SICS Technical Report
T2010:08
ISSN:1100-3154

Securing Internet of Things with Lightweight IPsec

Shahid Raza[1], Tony Chung[2], Simon Duquennoy[1], Dogan Yazar[1], Thiemo Voigt[1], Utz Roedig[2]
[1]*Swedish Institute of Computer Science, Kista, Sweden*
{shahid, simonduq, dogan, thiemo}@sics.se
[2]*Lancaster University Computing Department, Lancaster, UK*
{a.chung, u.roedig}@lancaster.ac.uk

February 7, 2011

Abstract Real-world deployments of wireless sensor networks (WSNs) require secure communication. It is important that a receiver is able to verify that sensor data was generated by trusted nodes. In some cases it may also be necessary to encrypt sensor data in transit. Recently, WSNs and traditional IP networks are more tightly integrated using IPv6 and 6LoWPAN. Available IPv6 protocol stacks can use IPsec to secure data exchange. Thus, it is desirable to extend 6LoWPAN such that IPsec communication with IPv6 nodes is possible. It is beneficial to use IPsec because the existing end-points on the Internet do not need to be modified to communicate securely with the WSN. Moreover, using IPsec, true end-to-end security is implemented and the need for a trustworthy gateway is removed.

In this paper we provide End-to-End (E2E) secure communication between an IP enabled sensor nodes and a device on traditional Internet. This is the first compressed lightweight design, implementation, and evaluation of 6LoWPAN extension for IPsec on Contiki. Our extension supports both IPsec's Authentication Header (AH) and Encapsulation Security Payload (ESP). Thus, communication endpoints are able to authenticate, encrypt and check the integrity of messages using standardized and established IPv6 mechanisms.

Keywords IPsec, 6LoWPAN, Contiki, IPv6, IoT

Acknowledge

This work has been performed within the SICS Center for Networked Systems funded by VINNOVA, SSF, KKS, ABB, Ericsson, Saab Systems, TeliaSonera, T2Data, Vendolocus and Peerialism. This work has been partially supported by CONET, the Cooperating Objects Network of Excellence.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Wireless Sensor Networks can be tightly integrated with existing IP based infrastructures using IPv6 over Low Power Wireless Personal Area Networks (6LoWPAN). Sensor nodes using 6LoWPAN can directly communicate with IPv6 enabled hosts and, for example, sensor data processing can be performed by standard servers. Thus, 6LoWPAN greatly simplifies operation and integration of WSNs in existing IT infrastructures.

1.1 Need for Security

Real-world deployments of wireless sensor networks (WSNs) require secure communication. Assume a (for this purpose simplified) Smart Metering application where a sensor network using IPv6 is used [1]. In this scenario, smart meters and the utility provider's computers need to securely communicate over paths that span a wireless network and the wired Internet. In particular, messages from the smart meter must be authenticated so that the utility provider can verify that the sensor reading was generated by the right smart meter, e.g. to avoid sending wrong invoices, and the smart meter must be able to verify that a command is actually from the right provider. It is also desirable to encrypt the sensor data in transit to ensure message confidentiality. IPv6 hosts in the Internet support by default IPsec for secure communication. Therefore, if data flows between IPv6 hosts and 6LoWPAN sensor nodes it is desirable to take advantage of existing capabilities and to secure traffic using IPsec. Thus, we propose to add IPsec support to 6LoWPAN as shown in Figure 1.1.

1.2 Research Contribution

IPsec defines an Authentication Header (AH) and an Encapsulating Security Payload (ESP). The AH can be used to provide data integrity and authentication while ESP provides data confidentiality, integrity and authentication. Either AH, ESP or both can be used to secure IPv6 packets in transit. It is up to

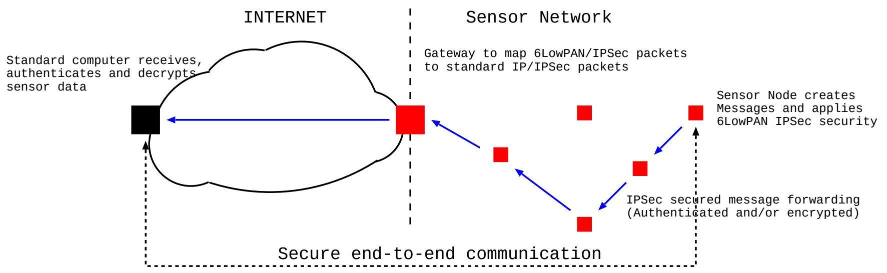

INTERNET
Sensor Network
Standard computer receives, authenticates and decrypts sensor data
Gateway to map 6LowPAN/IPSec packets to standard IP/IPSec packets
Sensor Node creates Messages and applies 6LowPAN IPSec security
IPSec secured message forwarding (Authenticated and/or encrypted)
Secure end-to-end communication

Figure 1.1: We propose to use IPsec to implement secure the communication between sensor nodes and hosts in an IPv6-enabled Internet. IPsec provides E2E security using existing methods and infrastructures.

the application to specify which security services are required. 6LoWPAN uses header compression techniques to ensure that the large IPv6 and transport-layer headers (UDP/TCP) are reduced. By supporting IPsec's AH an ESP additional IPv6 extension headers have to be included in each packet. Thus, it is important to ensure that compression techniques are as well applied to these extension headers. In case of ESP it is also necessary to find ways to allow header compression of the encrypted transport-layer header.

Independent of the achieved compression rates of AH and ESP it is obvious that IPsec support in 6LoWPAN will increase packet sizes as additional headers must be included. Note, however, that by using IPsec we do not need to use existing 802.15.4 link-layer security mechanisms which in turn frees some header space.

The main contributions of this paper are:

- *6LoWPAN-IPsec Specification:* We give a specification of IPsec for 6LoWPAN including definitions for AH and ESP extension headers. Prior to this work no specification for IPsec in the context of 6LoWPAN existed.
- *6LoWPAN-IPsec Implementation:* We present the first implementation of IPsec for 6LowPAN networks. We show that it is practical and feasible to secure WSN communication using IPsec.
- *6LoWPAN-IPsec Evaluation:* We evaluate the performance of our IPsec 6LoWPAN implementation in terms of code size, packet overheads and communication performance. Our results show that overheads are comparable to overheads of generally employed 802.15.4 link-layer security while offering the benefit of true E2E security.

The paper proceeds as follows. The next section discusses related work. Section 2.2 gives an overview of the different solutions for securing IP communications and motivates the use of IPsec. In Section 3.1 we present necessary background knowledge: IPv6, IPsec and 6LoWPAN. Section 4.1 describes the integration of 6LoWPAN and IPsec. In Section 5.1 we present our results and the subsequent section concludes the paper.

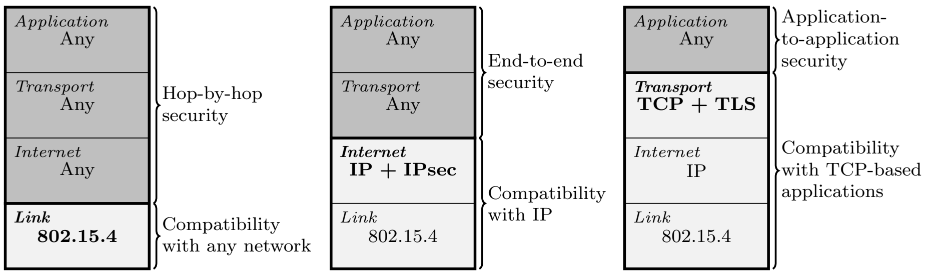
Chapter 2

Related Work

2.1 Related Work

Message authentication and encryption in WSNs is generally performed using well known cryptographic mechanisms such as block ciphers as part of standards-based protocols such as IEEE 802.15.4. However, these mechanisms are difficult to implement on resource constrained sensor nodes as cryptographic mechanisms can be expensive in terms of code size and processing speed. Furthermore, it is necessary to distribute and maintain keys and it is difficult to implement efficient key distribution protocols for resource constraint sensor nodes. Thus, a lot of research work aims to reduce complexity of cryptographic mechanisms, for example, TinyEEC [2] and NanoEEC [3], or to simplify key distribution, for example, Liu and Ning's proposal for pairwise key predistribution [4] and DHB-KEY [5]. These improvements make cryptographic mechanisms in the context of WSNs more viable but an important issue remains: a standardized way of implementing security services is missing and for each deployment unique customized solutions are created. Using the standardized 6LoWPAN as vehicle to implement security services in form of the proven and standardized IPsec offers a solution to this problem. IPsec is currently available as part of some WSN products, but does not provide a full E2E security solution. One such example is the ArchRock PhyNET [6] that applies IPsec in tunnel mode between the gateway and Internet hosts, but still relies on link-layer security within the sensor network thus breaking true E2E assurance. We are not aware of a complete E2E implementation nor an evaluation of a working system which we present in this paper.

The IEEE 802.15.4 [7] standard defines Advanced Encryption Standard (AES) message encryption and authentication on the link-layer. The cryptographic algorithms could be executed by specialized hardware within the transceiver chip. However, link-layer security only protects messages while they travel from one hop to the next as we discuss in Section 2.2. Wood and Stankovic [8] as well as Hu et al. [9] have demonstrated performance gains when



(a) Link-layer security (b) Internet-layer security (c) Transport-layer security

Figure 2.1: Communications can be secured (encrypted, checked and authenticated) at different layers of the protocol stack. Each solution involves its own scope and level of interoperability.

security operations are performed in hardware. We expect similar performance gains when IPsec operations are implemented in hardware.

Granjal et al. argue that IPsec is generally feasible in the context of WSN [10]. In their study they analyze the execution times and memory requirements of cryptographic algorithms. Their work only discusses performance of cryptographic algorithms but does not describe how IPsec is actually integrated with 6LoWPAN. In our work, we implement 6LoWPAN with compressed IPsec and we analyze the performance of the overall system and not only the performance of the cryptographic algorithms.

2.2 Securing Communications in the IP Stack

Researchers have unanimous consensus that security is very important for the future IP based WSN and its integration with the traditional Internet. IPv6 with potentially unlimited address space is the obvious choice for these networks [1]. However, security support for IP-based low power networks is still an open issue, as mentioned in the 6LoWPAN specifications [11, 12]. In fact, security can be guaranteed at different layers of the IP protocol stack, resulting in solutions with various compromises as illustrated by Figure 2.1.

6LoWPAN today relies on the IEEE 802.15.4 (referred to as 802.15.4 in the following) link-layer which provides data encryption and integrity checking. This solution is appealing since it is independent of the network protocols and is currently supported by the hardware of 802.15.4 radio chips. However, such link-layer mechanism only ensures *hop-by-hop* security where every node in the communication path (including the 6LoWPAN gateway) has to be be trusted, and where no host authentication neither keys management is supported. Furthermore, messages leaving the sensor network and continuing to travel on an IP network are not protected by link-layer security mechanism.

End-to-end security can be provided by the widely used Transport Layer

Security (TLS) standard. By operating between the transport-layer and the application-layer, it guarantees security between applications, includes a key exchange mechanism and provides authentication between Internet hosts in addition to confidentiality and integrity. As a counterpart, TLS can only be used over TCP, which is rarely used in wireless sensor networks. An adaptation of TLS for UDP called DTLS is however available, but it is not widely used and would require to be implemented in conjunction with TLS in order to provide full IP support, involving an important complexity overhead.

The IPsec protocol suite, mandated by IPv6, provides end-to-end security for any IP communication [13]. Like TLS and unlike hop-by-hop 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 protocols, including potential future ones. Furthermore, it ensures the confidentiality and integrity of the transport-layer headers (as well as the integrity of IP headers), which can not be done with a higher-level solution like TLS. For these reasons, researchers [10, 14, 15] and 6LoWPAN standardizations groups [12] consider IPsec a potential security solution for IP based WSN.

In this paper we show that a compressed IPsec is a sensible and viable choice for 6LoWPANs. The key advantage of using IPsec in WSN is that we achieve *end-to-end* IP based communication between a sensor device and Internet hosts. While using IPsec, the IEEE 802.15.4 security features can be disabled as security services are provided in the IP layer. We show later that when comparing link-layer security with IPsec, packet sizes are similar.

We leave to future work a general study on the trade-offs achieved depending on the layer at which the security is ensured. We plan an extensive comparison of link-layer security, IPsec and TLS that will allow us to understand the impact of each approach on performance, memory footprint, energy usage and packet sizes.

Chapter 3

Background

3.1 Background

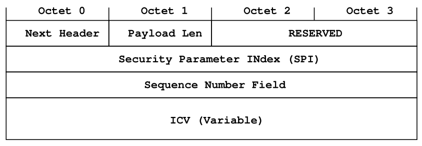
In this section we briefly outline core functionality of IPv6, IPsec and 6LoWPAN that is relevant for the work presented in this paper. For more information we refer to the corresponding RFCs: RFC2460 [16], RFC4301 [17] and RFC4944 [12].

3.1.1 IPv6 and IPsec

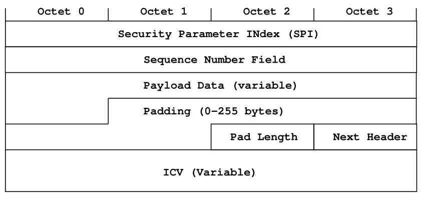
With the vision of the Internet of Things and Smart Objects all kind of physical devices such as wireless sensors are expected to be connected to the Internet via IP [1]. This requires the use of IPv6 [16], a new version of the Internet Protocol that increases the address size from $32bit$ to $128bit$. Besides the increased address space IPv6 provides in comparison to IPv4 a simplified header format, improved support for extensions and options, flow labeling capability and authentication and privacy capabilities.

Authentication and privacy in IPv6 is provided by IPsec [17]. IPsec defines a set of protocols for securing IP communication: the security protocols Authentication Header (AH) [18] and Encapsulating Security Payload (ESP) [19], the algorithms for authentication and encryption, key exchange mechanisms and so called security associations (SA) [17]. An SA specifies how a particular IP flow should be treated in terms of security. An SA is uniquely identified by a Security Parameter Index (SPI), an IP Destination Address, and a security protocol (AH or ESP) identifier. An SA bundles algorithms and parameters that are being used to secure a particular IP flow. A node stores SAs in the security association database (SADB). For each incoming or outgoing IP packet IPsec queries the SADB using SPI and IP destination address to obtain the SAs and to decide how to process the packet. How IPsec treats each packet is defined in the Security Policy Database (SPD).

The task of the AH is to provide connectionless integrity and data origin authentication for IP datagrams and protection against replays. A keyed Message Authentication Code (MAC) is used to produce authentication data. The



(a) Authentication Header (AH)



(b) Encapsulation Security Payload (ESP)

Figure 3.1: IPsec AH and ESP headers

MAC is applied to the IP header, AH header and IP payload. As some fields in the IP header may change while the packet is in transit these fields are set to zero for authentication data calculation, for example, Type of Service, TTL and Header Checksum. In the AH the authentication data field is set to zero for the calculation. The authentication header is shown in Figure 3.1a. It includes a reference to the next header (for example, TCP or UDP), a length field, the SPI that identifies the SA used, a sequence number to prevent replay attacks and the Integrity Check Value (ICV). The latter must be an integral multiple of $32bit$ for IPv6 in length and may therefore include explicit padding. All hosts must support at least the hash-based message authentication code algorithm HMAC-SHA1-96 [20] to calculate authentication data that has a size of $12byte$. Thus, as shown in Figure 3.1a, a basic AH header has a size of $24byte$.

ESP, shown in Figure 3.1b, provides origin authenticity, integrity, and confidentiality protection of IP packets. ESP is used to encrypt the payload of an IP packet but in contrast to AH it does not secure the IP header. If ESP is applied the IP header is followed by the ESP IP extension header which contains the encrypted payload. ESP includes an SPI that identifies the SA used, a sequence number to prevent replay attacks, the encrypted payload, padding which may be required by some block ciphers, a reference to the next header and optional authentication data. Encryption in ESP includes Payload Data, Padding, Pad Length and Next Header;Authentication, if selected, includes all header fields in the ESP. If we assume mandatory AES-CBC as encryption algorithm an ESP with perfect block alignment will have an overhead of $18byte$ (10 bytes for ESP and 8 bytes for Initialization Vector). If additional authentication using HMAC-SHA1-96 is used the ESP overhead is $30byte$, as the minimum length of HMAC-SHA1-96 is 12 bytes.

The protocols AH and ESP support two different modes: transport mode and tunnel mode. In transport mode 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 seems not practical as the additional headers would further increase the packet size.

3.1.2 6LoWPAN

6LoWPAN [12] aims at integrating existing IP based infrastructures and sensor networks by specifying how IPv6 packets are to be transmitted over an IEEE 802.15.4 network. The maximum physical-layer packet size of 802.15.4 packet is 127*byte* and the maximum frame header size is 25*byte*. An IPv6 packet has therefore to fit in 102*byte*. Given that packet headers of a packet would already consume 48*byte* of the available 102*byte* it is obvious that header compression mechanisms are an essential component of the 6LoWPAN standard.

HC13[21] proposes context aware header compression mechanisms: the LOWPAN_IPHC (referred to as IPHC in the following) encoding for IPv6 header compression and the LOWPAN_NHC (referred to as NHC in the following) encoding for the next header compression. The IPHC header is shown in Figure 3.2.

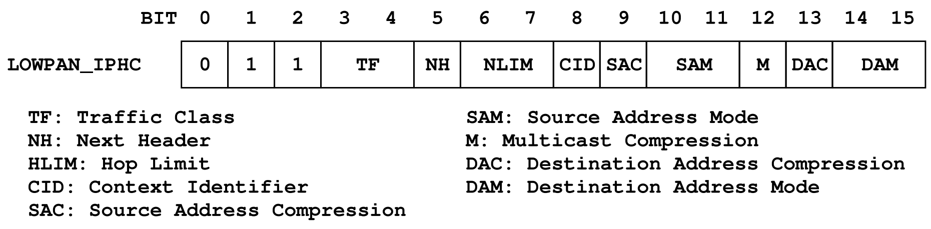


Figure 3.2: The LOWPAN_IPHC Header.

For efficient IPv6 header compression, IPHC removes safely IPv6 header fields that are implicitly known to all nodes in the 6LoWPAN network: Version is 6; Traffic Class and Flow Label are both zero; Payload Length is inferred from the 802.15.4 header; Hop Limit is set to a well-known value; addresses are formed using a single prefix and 802.15.4 addresses. The IPHC has a length of 2*byte* of which 13*bit* are used for header compression as shown in Figure 3.2. Uncompressed IPv6 header fields follow directly the IPHC encoding in the same order as they would appear in the normal IPv6 header.

In a multihop scenario IPHC can compress the IPv6 header to 7*byte* (2*byte* IPHC, 1*byte* Hop Limit, 2*byte* Source Address, and 2*byte* Destination Address). The NH field in the IPHC indicates whether the next header following the basic IPv6 header is encoded. If NH is 1, NHC is used to compress the next header. 6LoWPAN specifies that the size of NHC should be multiple of octets, usually 1*byte* where first variable length bits represents a NHC ID and the remaining bits are used to encode/compress headers. 6LoWPAN already defines NHC for UDP and IP Extension Header [21].

Chapter 4

IPsec Compression

4.1 6LoWPAN and IPsec

IPsec requires header compression to keep packet sizes reasonable in 6LoWPAN. Unfortunately, there are no header encodings specified for AH and ESP extension headers. In this section we therefore propose these extension header encodings. We evaluate our savings in terms of packet size later in Section 5.1. At the end of this section, we also discuss further improvements that would be possible by small, standard-complaint modifications of the end hosts where there is need for cryptographic algorithms that could handle 6LoWPAN UDP compression.

4.1.1 LOWPAN_NHC Extension Header Encoding

As previously described, HC13 defines context aware header compression using IPHC for IP header compression and NHC for the next header compression. The already defined NHC encoding form for IP extension headers can be used to encode AH and ESP extension headers. NHC encodings for the IPv6 Extension Headers consist of a NHC octet where three bits (bits 4,5,6) are used to encode the IPv6 Extension Header ID (EID). This NHC_EH encoding for extension headers is shown in Figure 4.1.

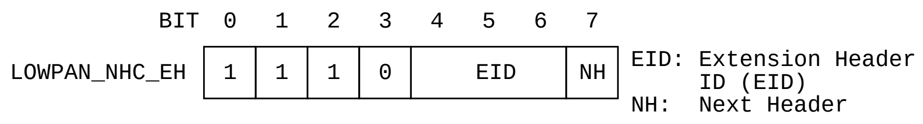


Figure 4.1: LOWPAN_NHC_EH: NHC encoding for IPv6 Extension Header

Out of eight possible values for the EID, six are specified by the HC13 draft. The remaining two slots (101 and 110) are currently reserved. We propose to use the two free slots to encode AH and ESP. Also, it is necessary to set the last bit in IPv6 extension header encoding to 1 to specify that the next header (AH or ESP) is encoded as well using NHC.

4.1.2 LOWPAN_NHC_AH Encoding

We define the NHC encoding for the AH. Our proposed NHC for AH is shown in Figure 4.2.

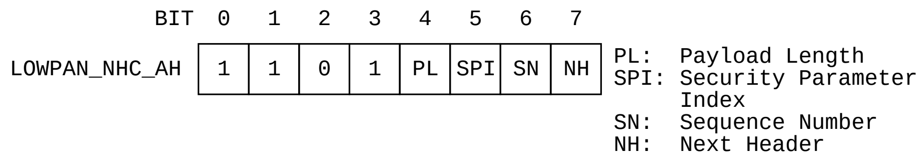


Figure 4.2: NHC_AH: NHC encoding for IPv6 Authentication Header

We describe the function of each header field:

- The first four bits in the NHC_AH represent the NHC ID we define for AH. These are set to 1101. We do not necessarily need these four bits because the EID field of the previous NHC identifies the next header that will be either AH or ESP. However to comply with 6LoWPAN standard we set these ID bits accordingly.

- 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 depend 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.

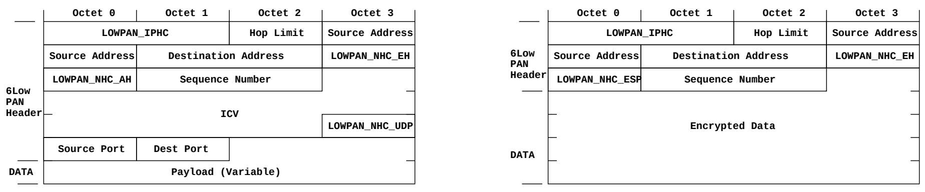
- If $SPI = 0$: the default SPI for the sensor network is used and the SPI field is omitted. We set the default SPI value to 1. SPI 0 is reserved to indicate that no security association exists. This does not mean that all nodes use the same security association (SA), but that every node has a single preferred SA, identified by SPI 1.

  If $SPI = 1$: All 32 bits indicating the SPI are carried inline after the NHC_AH header.

- If $SN = 0$: A 16 bit sequence number is used. The left most 16 bits are assumed to be zero.

  If $SN = 1$: All 32 bits of the sequence number are carried inline after the NHC_AH header.

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



(a) AH compression (b) ESP compression

Figure 4.3: Example of a compressed IPv6/UDP packet using AH or ESP

If $NH = 1$: The next header field in AH is skipped. The next header will be encoded using NHC.

The minimum length of a standard AH supporting the mandatory HMAC-SHA1-96 is $24 byte$. After optimal compression we obtain a header size of $16 byte$. If we define an additional algorithm called HMAC-SHA1-32 that uses only the most significant $32 byte$ of HMAC-SHA1-96 as ICV the AH size of $16 byte$ can be reduced to $8 byte$. Figure 4.3a and 4.3b shows compressed IPv6/UDP packet secured with AH and EPS with HMAC-SHA1-96, respectively.

4.1.3 LOWPAN_NHC_ESP Encoding

Figure 4.4 shows the NHC encodings we propose for ESP.

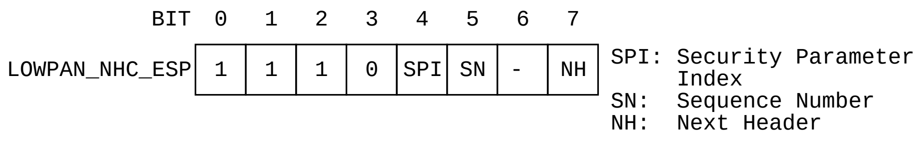


Figure 4.4: LOWPAN_NHC_ESP: NHC encoding for IPv6 ESP

We describe the function of each header field:

- The first 4 bits in the NHC_ESP represent the NHC ID we define for ESP. These are set to 1110.
- If $SPI = 0$: The default SPI for the sensor network is used and the SPI field is omitted. We set the default SPI value to 0.

  If $SPI = 1$: All 32 bits indicating the SPI are carried inline after the NHC_ESP header.
- If $SN = 0$: A 16 bit sequence number is used. The left most 16 bits are assumed to be zero.

If $SN = 1$: All 32 bits of the sequence number are carried inline after the NHC_ESP header.

- 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 field in ESP is skipped. The next header will be encoded using NHC. This is only possible if hosts are able to execute 6LoWPAN compression/decompression and encryption/decryption jointly.

Recall that the minimum ESP overhead without authentication, AES-CBC and perfect block alignment is $18byte$. After optimal compression this header overhead is reduced to $12byte$. ESP with authentication (HMAC-SHA1-96) has an overhead of $30byte$ which is reduced to $24byte$ using the outlined ESP compression. To optimize the authentication option in ESP the standard algorithms such as HMAC-SHA1-96 could be replaced with HMAC-SHA1-32.

4.1.4 Combined Usage of AH and ESP

It is possible to use AH and ESP in combination; obviously the defined AH and ESP compression headers can be used in succession. However, it is more efficient in terms of header sizes to use ESP with authentication option than to apply AH and ESP to a packet. As packet sizes are important in the context of WSNs we expect that this IPsec option will not be used in practice.

4.1.5 End Host Requirement

AH capable 6LoWPAN nodes can directly communicate with unmodified IPsec hosts on conventional Internet. When ESP is used 6LoWPAN nodes can as well communicate directly with unmodified IPsec hosts. However, if ESP is used it is not possible to compress upper layer headers such as UDP. A 6LoWPAN gateway between sensor network and IP network cannot access and expand the encrypted UDP header. To enable UDP compression with ESP we need to specify a new encryption algorithm for ESP which is able to perform UDP header compression and encryption. Again, if this optimization is used IPsec hosts must include and support this encryption protocol.

Chapter 5

Evaluation and Results

5.1 Evaluation and Results

This section evaluates the viability of IPsec for IP-based WSN. After describing our implementation and experimental setup, we evaluate the impact of IPsec in terms of memory footprint, packet size, energy consumption and performances under different configurations.

5.1.1 Implementation and Experimental Setup

We implement IPsec AH and ESP for Contiki [22], an operating system developed for resource constrained devices. The implementation required the modification of the existing Contiki μIP stack which already provides 6LoWPAN functionality. The Contiki μIP stack is used on the sensor nodes and on a so called soft bridge connecting WSN and the Internet. In addition to the IPsec protocol, we implement the IPsec/6LoWPAN compression mechanisms as outlined in the previous section. We support the NHC_EH, NHC_AH, and NHC_ESP encodings (see Section 4.1) at the SICSLoWPAN layer, the 6LoWPAN component of the μIP stack. We use the SHA1 and AES implementations

Service	Uncompressed IPsec		Compressed IPsec		802.15.4	
	Mode	Bytes	Mode	Bytes	Mode	Bytes
AH Authentication	HMAC-SHA1-96	24	HMAC-SHA1-96	16	AES-CBC-MAC-96	12
ESP Encryption	AES-CBC	18	AES-CBC	12	AES-CTR	5
ESP Encryption and Authentication	AES-CBC and HMAC-SHA1-96	30	AES-CBC and HMAC-SHA1-96	24	AES-CCM-128	21

Table 5.1: With compressed IPsec, packet sizes are similar to 802.15.4 while IPsec provides end-to-end security.

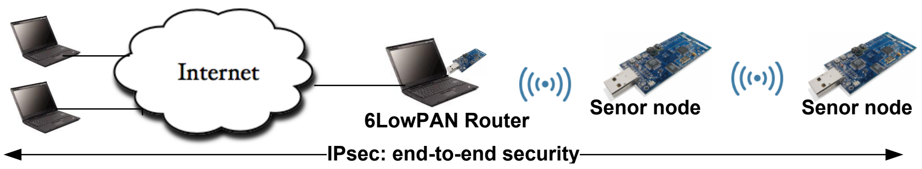


Figure 5.1: Multihop testbed setup for our Experiments and Evaluation.

from MIRACL [23], an open source library, and implement all cryptographic modes of operation needed for authentication and encryption in IPsec. 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. One of the reasons for implementing both AES-CTR and AES-CBC is that we can later combine them to use in AES Counter with CBC-MAC (CCM) mode [24], which is used in 802.15.4 link-layer security; we may later take advantage of an hardware-based AES-CCM implementation for IPsec AH authentication and/or ESP encryption and authentication.

At this point in time, our Contiki IPsec 6LoWPAN implementation does not support key exchange mechanisms such as the Internet Key Exchange (IKE) protocol. Keys are set manually before deployment. However, it has to be noted that manual key distribution is currently used as well for traditional 802.15.4 link-layer security. In future work, we intend to add support for key distribution protocols, to evaluate their efficiency in 6LoWPAN and to discuss parameters such as the frequency of key renewal according to given security requirements.

Our evaluation setup is illustrated in Figure 5.1. It consists of two Tmote Sky [25] sensor nodes that feature the IEEE 802.15.4 compliant CC2420 radio, a 6LoWPAN soft bridge (implemented by a third Tmote) as well as a Linux machine running Ubuntu OS with IPsec enabled. The two sensor nodes on the right side in the figure present a multihop network. They execute a single application which listens to a fixed UDP port. When a packet is received, it is processed by the 6LoWPAN layer, interpreted by the IPsec layer and by μIP, then its payload is forwarded to the application. As a reply, a new datagram of the same size is sent back, following the opposite process. Thus, IPsec is used to secure end-to-end communication between the 6LoWPAN node and the Internet host. In order to avoid the delay of a duty-cycled MAC layer, we use Contiki's NULLMAC MAC in the experiments and hence the nodes keep their radio turned on all the time.

5.1.2 Memory footprint

We measure the ROM and RAM footprint of our IPsec implementation. Table 5.2 compares IPsec AH and IPsec ESP using the multiple modes of operation

	ROM footprint (kB)		RAM footprint (kB)	
System	overall	overhead	overall	additional
Without IPsec	32.9	–	8.0	–
AH with HMAC-SHA1-96	36.8	3.9	9.1	1.1
AH with XCBC-MAC-96	38.4	5.5	8.5	0.5
ESP with AES-CBC	41.4	8.5	8.3	0.3
ESP with AES-CTR	39.8	6.9	9.1	0.3
ESP with AES-XCBC-MAC-96	39.8	6.9	8.3	0.3
ESP with AES-CBC + AES-XCBC-MAC-96	41.9	9.0	8.3	0.3
ESP with AES-CTR + AES-XCBC-MAC-96	40.3	7.4	8.3	0.3

Table 5.2: ROM and RAM footprints show that AH and ESP consumes just 3.9kB and 9kB, respectively, for mandatory IPsec algorithms

we implemented. The footprints are compared with a reference Contiki system including uIP and SICSLoWPAN.

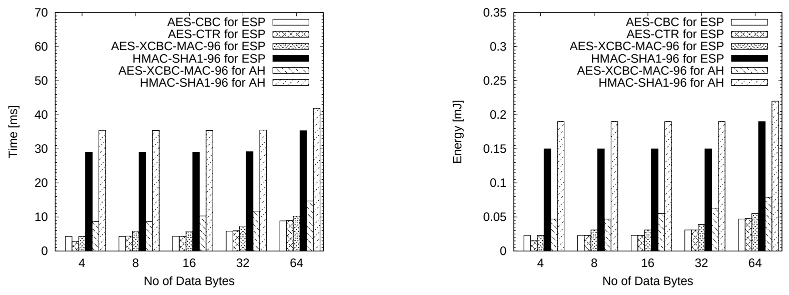
The ROM footprint overhead ranges from 3.8 kB (AH with HMAC-SHA1) to 9 kB (ESP with AES-CBC + AES-XCBC-MAC). This always keeps the system footprint under 48 kB, the Flash ROM size of the Tmote Sky. It is worth mentioning that unlike AES-CBC, the AES-CTR mode of operation only relies on AES encryption. Thus, the AES-CTR + AES-XCBC-MAC-96 configuration can be implemented without AES decryption, resulting in a particularly low memory footprint.

The RAM footprint is calculated as the sum of the global data and the runtime stack usage that we measure by running Contiki in the MSPim emulator [26]. 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 both IPsec AH and ESP can be embedded in constraint devices while leaving space for applications.

5.1.3 Packet Overhead Comparison

Currently WSN communication is secured using 802.15.4 link-layer security. This security mechanism can only provide hop-by-hop security and, in contrast to our IPsec implementation, lacks the ability to provide proper end-to-end-security. Nevertheless, we provide here a comparison of packet overheads between 802.15.4 link-layer security and IPsec security. Table 5.1 summarizes the packet overhead when using uncompressed IPsec, compressed IPsec and 802.15.4 link-layer security.

If only message authentication is required the 802.15.4 link-layer security provides three options: AES-CBC-MAC-32, AES-CBC-MAC-64 and AES-CBC-MAC-128, where the last digit indicates the Message Authentication Code (MAC) length in bits. This MAC follows in 802.15.4 directly after the payload and no extra header fields are used. Thus the packet overhead for the



(a) Completion time (b) Energy consumption

Figure 5.2: The comparison of our implemented algorithms shows that among the ones specified in the standards, AES-CBC and AES-XCBC-MAC-96 are the most efficient in terms of processing time and energy consumption. They are also mandatory and the most secure.

authentication scheme is exactly the length of the MAC. In IPsec we can use AH to provide authentication only. The available HMAC-SHA1-96 provides a MAC code of 12*byte* length. The additional AH header fields increase the overhead to 24*byte*. Thanks to the IPsec header compression we defined, this overhead is reduced to 16*byte*. In the case of HMAC-SHA1-32, 802.15.4 has a 4*byte* overhead whilst our compressed IPsec provides a 8*byte* overhead. The ability to provide end-to-end authentication with IPsec has hence a cost of 4*byte* compared to the 802.15.4 baseline which provides only hop-by-hop security.

If only message encryption is required, the 802.15.4 link-layer security provides AES-CTR which has a 5*byte* overhead. In comparison, IPsec with ESP and AES-CBC leads to an overhead of 18*byte*, reduced to 12*byte* thanks to header compression. Here, the ability to provide end-to-end encryption with IPsec has a cost of 7*byte* compared to the 802.15.4 baseline.

If both encryption and authentication are required, the 802.15.4 link-layer security provides three versions of AES-CCM: AES-CCM-32, AES-CCM-64, AES-CCM-128. These versions differ in MAC size and have packet overheads of 9, 13 and 21*byte*. ESP with AES-CBC and HMAC-SHA1-96 can provide combined authentication and encryption. With AES-CCM-128, the overhead for 802.15.4 is 21*byte* while IPsec has an overhead of 30*byte*, reduced to 24*byte* when using our 6LoWPAN compression extension. The ability to provide end-to-end encryption and authentication with IPsec has hence a cost of 3*byte* compared to the 802.15.4 baseline.

5.1.4 Performance of Cryptography

We evaluate the efficiency of the different cryptographic algorithms and modes supported by our IPsec implementation. Figure 5.2 details the performances and energy consumption for each mode of operation and depending on the size

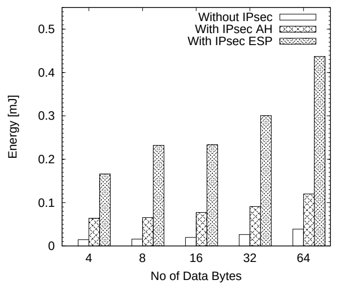


Figure 5.3: Node energy consumption is lower without IPsec and higher for ESP than for AH. Compared to other activities e.g. idle listening it is not significant.

of the IP payload. The authentication algorithms are compared separately for AH and ESP: with AH the MAC is calculated over the IP header and payload packet, while in ESP the IP header is neither encrypted nor authenticated.

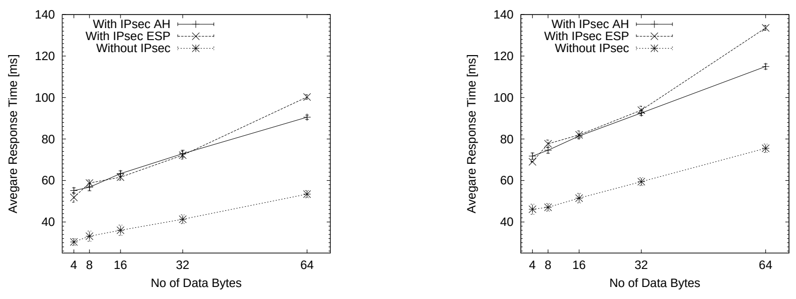
Our results show that for encryption, AES-CBC and AES-CTR have similar performances and energy consumption. Regarding authentication, the cost is as expected higher for AH than for ESP because of the processing of the $40\mathit{byte}$ IP header. In all cases, the energy consumption has a fixed-cost and grows linearly with the data size. HMAC-SHA1-96 is not as efficient as other solutions because of its particularly important fixed-cost when data sizes are small. On larger data sizes, its relative energy overhead would decrease.

The proposed standard for Cryptographic Suites for IPsec [27] specifies that the future IPsec systems will use AES-CBC-128 for encryption and AES-XCBC-MAC-96 mode for authentication. As these are also among the efficient algorithms we use these cryptographic algorithms in the following experiments.

5.1.5 System-wide Energy Overhead

Energy is a precious resource in sensor networking and hence it is important to minimize its consumption. We measure the energy overhead of IPsec AH and ESP on the Tmote Sky using Contiki's real-time timers. When a node receives a packet we begin to count the CPU ticks when starting AH decompression. We stop counting when the AH Compression is finished. Hence, we include 6LoWPAN decompression, μIP stack handling for incoming packets, application-layer processing, μIP stack processing for outgoing packets and 6LoWPAN compression. We repeat the experiments without IPsec.

Figure 5.3 shows the energy consumption of AH, ESP, and without using IPsec. The results are the average of 20 runs. The variance is very low and hence not shown. The results show that ESP consumes more energy than AH; this is because for ESP we use both authentication and encryption. Although the energy consumption with IPsec is higher than without IPsec we argue that this increase is more than compensated by the fact that we gain end-to-end security between a sensor node and nodes on the wired Internet.



(a) Single Hop

(b) Multi Hop

Figure 5.4: The difference between an IPsec enabled system and a system without IPsec is constant across a single hop and a multihop network.

5.1.6 System-wide Processing Time Overhead

We measure and evaluate the response time for different data sizes with IPsec (for both AH and ESP) and without IPsec. In our experiment the response time is the time it takes to send a message from an IP connected Linux machine to a sensor node and to receive a response.

Figure 5.4 shows the response time of IP packets with AH, ESP, and no IPsec for single hop and multiple hop (two hops) 6LoWPANs. The figure shows that both AH and ESP have similar response times for small packets. However, the response time for ESP increases as the packet size increases. This is because the AH is calculated on the entire IP header but for authentication only, while ESP secures the IP payload only but with both encryption and authentication.

We observe that the difference between the response time with IPsec and without IPsec remains constant when we compare single and multi-hop results. The reason for this is that the forwarding sensor nodes do not need to do any IPsec processing. However, if the size of data increases, the IP packet is fragmented to fit in the 127 bytes 802.15.4 frames, which increases the system-wide response time. This is apparent in Figure 5.4b where, for data size of 64 bytes, the difference between single hop and multihop networks is not constant for ESP.

Summary

Our evaluation shows that the IPsec AH and ESP fit in a tiny sensor node (e.g. Tmote Sky) with still room available for applications. Our cryptographic algorithms analysis show that our implementations for AES-CBC and AES-XBC-MAC-96 –the IPsec standard recommended algorithms for future Internet– are faster in terms of processing time and efficient regarding energy consumption and could be definitely used in 6LoWPAN realm. The system-wide response time comparisons with and without IPsec shows that IPsec has a bit higher response time compared with no IPsec. Also, IPsec consumes a bit more energy but the overhead is not high compared to other activities such as idle listening that in many cases dominates the power consumption [28].

Chapter 6

Conclusions and Future Work

In this paper we have given a specification of IPsec for 6LoWPAN. Furthermore we have presented an implementation of IPsec for 6LowPAN and we have demonstrated that it is possible and feasible to use this mechanism to secure communication between sensor nodes and hosts in the Internet.

WSNs will be an integral part of the Internet of the future. Communication between hosts and nodes in sensor networks will be commonplace. The research community and industry agrees that IP and 6LoWPAN are the protocol standards that will be used to bring the Internet and WSNs together. IPsec is the standard method to secure IP communication and it is therefore reasonable to investigate if this mechanism can be extended to reach nodes within the WSN. The work presented in this paper shows that IPsec is indeed feasible in the context of 6LoWPAN.

In our current implementation we set security keys manually. In real deployments, however, key exchange should be automatic. As a future work we intend to develop IPsec's Internet Key Exchange (IKE) for 6LoWPANs. Also, we intend to develop TLS as application level solution and thoroughly compare it with IPsec.

Bibliography

[1] J. Vasseur and A. Dunkels. *Interconnecting Smart Objects with IP - The Next Internet*. Morgan Kaufmann, 2010.

[2] A. Liu and P. Ning. TinyECC: A configurable library for elliptic curve cryptography in wireless sensor networks. In *IPSN 2008*, Washington, DC, USA, 2008.

[3] P. Szczechowiak, L. Oliveira, M. Scott, M. Collier, and R. Dahab. Nanoecc: Testing the limits of elliptic curve cryptography in sensor networks. In *EWSN 2008*, February 2008.

[4] D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. In *10th ACM conference on Computer and communications security (CCS)*, New York, NY, USA, 2003.

[5] A. Chung and U. Roedig. DHB-KEY: An Efficient Key Distribution Scheme for Wireless Sensor Networks. In *WSNS2008*, Atlanta, USA, September 2008.

[6] ArchRock Corporation. Phynet n4x series, 2008.

[7] IEEE Computer Society. Ieee std. 802.15.4-2006, 2006.

[8] A. Wood and J. Stankovic. Poster abstract: AMSecure - secure link-layer communication in TinyOS for IEEE 802.15.4-based wireless sensor networks. In *ACM SenSys*, Boulder, USA, November 2006.

[9] W. Hu, P. Corke, W. Shih, and L. Overs. secfleck: A public key technology platform for wireless sensor networks. In *EWSN 2009*, Cork, Ireland, February 2009.

[10] J. Granjal, R. Silva, E. Monteiro, J. Sa Silva, and F. Boavida. Why is IPsec a viable option for wireless sensor networks . In *WSNS2008*, Atlanta, USA, September 2008.

[11] N. Kushalnagar, G. Montenegro, and C. Schumacher. IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals. RFC 4919, August 2007.

[12] G. Deloche, N. Kushalnagar, J. Hui, and D. Culler. Transmission of IPv6 Packets over IEEE 802.15.4 Networks. RFC 4944, September 2007.

[13] S. Kent and R. Atkinson. Security architecture for the internet protocol, 1998.

[14] R. Riaz, Ki-Hyung Kim, and H.F. Ahmed. Security analysis survey and framework design for ip connected lowpans. In *ISADS '09*, mar. 2009.

[15] R. Roman and J. Lopez. Integrating wireless sensor networks and the internet: a security analysis. *Internet Research*, 19(2):246–259, 2009.

[16] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460, December 1998.

[17] S. Kent and K. Seo. Security architecture for the internet protocol. RFC 4301, 2005.

[18] S. Kent. IP Authentication Header. RFC 4302, 2005.

[19] S. Kent. Ip encapsulating security payload (esp). RFC 4303, 2005.

[20] V. Manral. Cryptographic algorithm implementation requirements for encapsulating security payload (esp) and authentication header (ah). RFC 4835, 2007.

[21] J. Hui and P. Thubert. Compression Format for IPv6 Datagrams in 6LoWPAN Networks. draft-ietf-6lowpan-hc-13, September 2010.

[22] A. Dunkels, B. Grönvall, and T. Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *EMNets'04*, Tampa, USA, November 2004.

[23] Shamus Software. Multiprecision Integer and Rational Arithmetic C/C++ Library. Web page. Visited 2010-04-17.

[24] D. Whiting, R. Housley, and N. Ferguson. Counter with CBC-MAC (CCM). RFC 3610 (Informational), September 2003.

[25] J. Polastre, R. Szewczyk, and D. Culler. Telos: enabling ultra-low power wireless research. In *IPSN'05*, apr. 2005.

[26] J. Eriksson, A. Dunkels, N. Finne, F. Österlind, and T. Voigt. Mspsim – an extensible simulator for msp430-equipped sensor boards. In *Proceedings of the European Conference on Wireless Sensor Networks (EWSN), Poster/Demo session*, Delft, The Netherlands, January 2007.

[27] P. Hoffman. Cryptographic Suites for IPsec. RFC 4308, December 2005.

[28] A. Dunkels, F. Österlind, N. Tsiftes, and Z. He. Demo abstract: Software-based sensor node energy estimation. In *Proceedings of the Fifth ACM Conference on Networked Embedded Sensor Systems (SenSys 2007)*, Sydney, Australia, November 2007.