Securing Communication in 6LoWPAN with Compressed IPsec

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

***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. 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 IP enabled sensor networks and the traditional Internet. This is the first compressed lightweight design, implementation, and evaluation of 6LoWPAN extension for IPsec. 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.

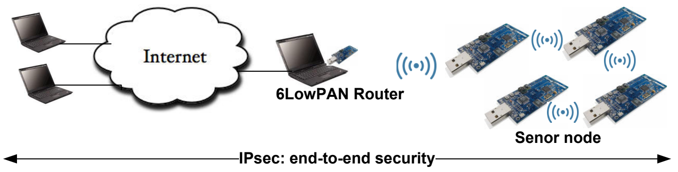


Fig. 1: We propose to use IPsec to secure the communication between sensor nodes in 6LoWPANs and hosts in an IPv6-enabled Internet. IPsec provides E2E security using existing methods and infrastructures.

I. 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.

Real-world deployments of wireless sensor networks (WSNs) require secure communication. For instance, in a smart meter application, the provider and the meters would need to authenticate one another and encryption would be desirable to ensure data 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 illustrated by Figure 1.

IPsec defines an Authentication Header (AH) and an Encapsulating Security Payload (ESP). The AH provides 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 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 and ESP, additional IPv6 extension headers have to be included in each datagram. Thus, it is important to ensure that compression techniques are as well applied to these extension headers.

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 by discussing related work followed by a further motivating of using of IPsec. Then we present background knowledge on IPv6, IPsec and 6LoWPAN. Section V describes our proposed integration of 6LoWPAN and IPsec. After a thorough experimental evaluation of the performance of our IPsec implementation, we conclude the paper.

II. 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 constrained sensor nodes. Thus, a lot of research work aims to reduce complexity of cryptographic mechanisms, for example, TinyEEC [?] and NanoEEC [?], or to simplify key distribution, for example, Liu and Ning's proposal for pairwise key predistribution [?] and DHB-KEY [?]. 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 a 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 [?] 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 [?] 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 III. Wood and Stankovic [?] as well as Hu et al. [?] have demonstrated performance gains when 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 [?]. 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, not only the performance of the cryptographic algorithms.

III. Securing WSN Communications

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 [?]. However, security support for IP-based low power networks is still an open issue, as mentioned in the 6LoWPAN specifications [?], [?]. Actually, security can be guaranteed at different layers of the IP protocol stack, resulting in solutions with various compromises..

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 neither host authentication nor key management is supported. Furthermore, messages leaving the sensor network and continuing to travel on an IP network are not protected by link-layer security mechanisms.

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 available, but it is not widely used.

The IPsec protocol suite, mandated by IPv6, provides end-to-end security for any IP communication [?]. 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 cannot be done with a higher-level solution like TLS. For these reasons, researchers [?], [?], [?] and 6LoWPAN standardizations groups [?] consider IPsec a potential security solution for IP based WSN.

In this paper we show that 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. When 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.

IV. Background

In this section we briefly outline core functionality of IPv6, IPsec and 6LoWPAN that is relevant for the work presented in

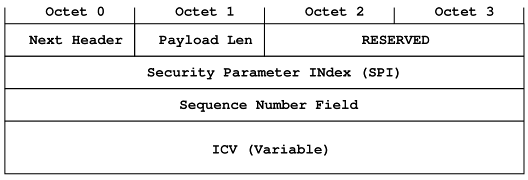


Fig. 2: IPsec AH headers

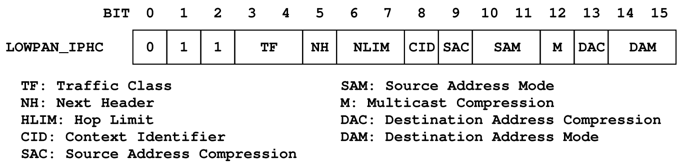

BIT 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
LOWPAN_IPHC
0 1 1 TF NH NLIM CID SAC SAM M DAC DAM
TF: Traffic Class
NH: Next Header
HLIM: Hop Limit
CID: Context Identifier
SAC: Source Address Compression
SAM: Source Address Mode
M: Multicast Compression
DAC: Destination Address Compression
DAM: Destination Address Mode

Fig. 3: The LOWPAN_IPHC Header.

this paper. For more information we refer to the corresponding RFCs: RFC2460 [?], RFC4301 [?] and RFC4944 [?].

A. *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 [?]. This requires the use of IPv6 [?], a new version of the Internet Protocol that increases the address size from 32 bits to 128 bits. 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 [?]. IPsec defines a set of protocols for securing IP communication: the security protocols Authentication Header (AH) [?] and Encapsulating Security Payload (ESP) [?], the algorithms for authentication and encryption, key exchange mechanisms and so called security associations (SA) [?]. An SA specifies how a particular IP flow should be treated in terms of security. To establish SAs, IPSec standard specifies both pre-shared key and Internet Key Exchange (IKE) protocol. This means that every node on IPv6 enabled conventional Internet supports pre-shared key. In other words an implementation with pre-shared based SA establishment works with any IPv6 node on Internet. Also, IKE uses asymmetric cryptography that is assumed to be heavy weight for small sensor nodes. However, it would be worth investigating IKE with ECC for 6LoWPANs; we intend to do it in future.

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 MAC is applied to the IP header, AH header and IP payload. The authentication header is shown in Figure 2. All hosts must support at least the hash-based message authentication code algorithm AES-XCBC-MAC-96 [?] to calculate authentication data that has a size of 12 bytes. Thus, as shown in Figure 2, a basic AH header has a size of 24 bytes.

ESP [?] 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 18 bytes (10 bytes for ESP and 8 bytes for Initialization Vector). If additional authentication using AES-XCBC-MAC-96 is used the ESP overhead is 30 bytes, as the minimum length of AES-XCBC-MAC-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.

B. *6LoWPAN*

6LoWPAN [?] 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 bytes and the maximum frame header size is 25 bytes. An IPv6 packet has therefore to fit in 102 bytes. Given that packet headers of a packet would already consume 48 bytes of the available 102 bytes it is obvious that header compression mechanisms are an essential component of the 6LoWPAN standard.

HC13[?] 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.

For efficient IPv6 header compression, IPHC removes safely IPv6 header fields that are implicitly known to all nodes in the 6LoWPAN network. The IPHC has a length of 2 byte of which 13 bits are used for header compression as shown in Figure 3. 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 bytes 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

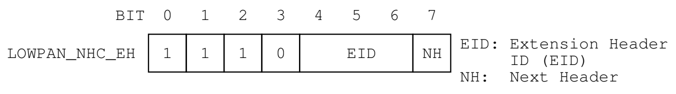


Fig. 4: LOWPAN_NHC_EH: NHC encoding for IPv6 Extension Header

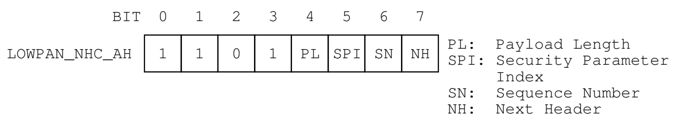


Fig. 5: NHC_AH: NHC encoding for IPv6 Authentication Header

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 [?].

V. 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 VI. At the end of this section, we also discuss further improvements that would be possible by small, standard-compliant modifications of the end hosts where there is need for cryptographic algorithms that could handle 6LoWPAN UDP compression.

A. 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.

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.

B. LOWPAN_NHC_AH Encoding

We define the NHC encoding for the AH. Our proposed NHC for AH is shown in Figure 5.

We describe the function of each header field:

- The first four bits in the NHC_AH represent the NHC ID we define for AH, and are set to 1101. These are needed to comply with 6loWPAN standard.

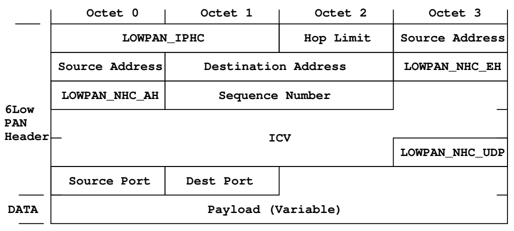


Fig. 6: Example of a compressed IPv6/UDP packet using AH

- PL: If 0, the payload lengths 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 1, the length is carried inline after the NHC_AH header
- SPI: If 0, the default SPI for the sensor 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 security association (SA), but that every node has its own preferred SA, identified by SPI 1. If 1, the SPI is carried inline
- SN: If 0, a 16 bit sequence number is used and the left most 16 bits are assumed to be zero. If 1, all 32 bits of the sequence number are carried inline.
- NH: If 0, the next header field in AH will be used to specify the next header and it is carried inline. If 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 bytes. After optimal compression we obtain a header size of 16 bytes. Figure 6 shows compressed IPv6/UDP packet secured with AH with HMAC-SHA1-96.

C. LOWPAN_NHC_ESP Encoding

Also the NHC encoding for ESP encodes the security parameter index, the sequence number, the next header fields and the NHC ID for ESP. In the case of ESP, we propose 1110 as NHC ID while we propose 1101 as NHC for AH as shown in Figure 6. Due to space limitation, we do not detail these encoding for ESP which are available in full details in a technical report [?].

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 12 bytes. ESP with authentication (HMAC-SHA1-96) has an overhead of 30 bytes which is reduced to 24 bytes using the outlined ESP compression.

D. 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

System	ROM (kB)		RAM (kB)	
	overall	diff	overall	diff
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-CBC + AES-XCBC-MAC-96	41.9	9.0	8.3	0.3

TABLE I: Memory footprints show that AH and ESP consumes just 3.9kB and 9kB for mandatory IPsec algorithms

context of WSNs we expect that this IPsec option will not be used in practice.

E. 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.

VI. Evaluation and Results

In this section we quantify performance of the proposed IPSec extensions for 6LowPAN. 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.

A. Implementation and Experimental Setup

We implement IPsec AH and ESP for the Contiki operating system [?]. 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 V) at the SICSLoWPAN layer, the 6LoWPAN component of the μIP stack.

We use the SHA1 and AES implementations from MIRACL [?], 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. Our Contiki IPsec 6LoWPAN implementation uses pre-shared keys to establish SAs which work with any IPv6 node on Internet as a pre-shared mechanism is mandatory in IPsec. Manual key distribution, however, is currently also used for traditional 802.15.4 link-layer security.

Our evaluation setup shown in Figure 1 consists of four Tmote Sky [?] sensor nodes, a 6LoWPAN soft bridge (implemented by a fifth Tmote) nd a Linux machine running Ubuntu OS with IPsec enabled. The four sensor nodes on the right side in Figure 1 form a multihop network. They execute a single application that 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 the end-to-end (E2E) communication between the sensor node and the Internet host. To avoid the delay of a duty-cycled MAC layer, we use Contiki's NullMAC in the experiments and hence all nodes keep their radio turned on all the time.

B. Memory footprint

We measure the ROM and RAM footprint of our IPsec implementation. Table I compares IPsec AH and IPsec ESP using the multiple modes of operation 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 MSPSim emulator [?]. 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 constrained devices while leaving space for applications.

C. 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 IPsec, lacks the ability to provide proper E2E security. Nevertheless, we provide here a comparison of packet overheads between 802.15.4 link-layer security and IPsec security. Table II summarizes the packet overhead when using uncompressed IPsec, compressed IPsec and 802.15.4 link-layer security.

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

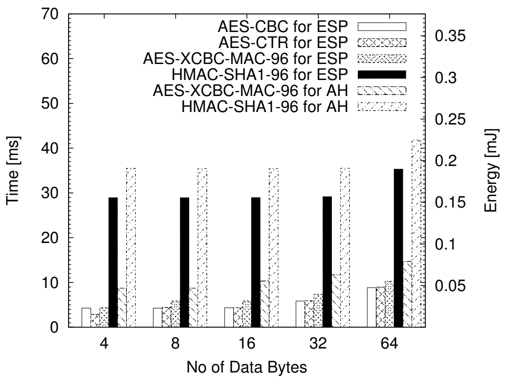


Fig. 7: The comparison of our implemented algorithms shows that among the ones specified in the standards, AES-CBC and AES-XCBC-MAC-96 are the most efficient in terms of processing time and energy consumption. They are also mandatory and the most secure.

When using link-layer security, the packet overhead for the authentication scheme is exactly the length of the MAC. In IPsec when using AES-XCBC-MAC-96, the MAC has a length of 12 bytes. The additional AH header fields increase the overhead to 24 bytes. Thanks to the IPsec header compression we defined, this overhead is reduced to 16 bytes. The ability to provide E2E authentication with IPsec has hence a cost of 4 bytes 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 bytes overhead. In comparison, IPsec with ESP and AES-CBC leads to an overhead of 18 bytes, reduced to 12 bytes thanks to header compression. Here, the ability to provide E2E encryption with IPsec has a cost of 7 bytes compared to the 802.15.4 baseline.

With AES-CCM-128, the overhead for 802.15.4 is 21 bytes while IPsec ESP has an overhead of 30 bytes, reduced to 24 bytes when using our 6LoWPAN compression extension. The ability to provide E2E encryption and authentication with IPsec has hence a cost of 3 bytes compared to the 802.15.4 baseline.

Moreover, when carrying large IP datagrams, link-layer fragmentation has to be used. With link-layer security, one pays the header overheads for every fragment. In contrast, the IPsec header is included only once for all the fragments of a single datagram. This means that as soon as two or more fragments are needed, IPsec offers a lower header overhead than 802.15.4 link-layer security.

D. Performance of Cryptography

We evaluate the efficiency of the different cryptographic algorithms and modes supported by our IPsec implementation. Figure 7 details the performances and energy consumption for each mode of operation and depending on the size 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.

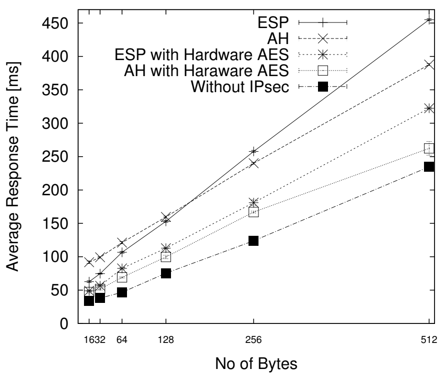
Our results show that for encryption, AES-CBC and AES-CTR have similar performances and energy consumption. Regarding authentication, the cost is as expected higher for AH than for ESP because of the processing of the 40 byte IP header. In all cases, the energy consumption has a fixed-cost and grows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

The proposed standard for Cryptographic Suites for IPsec specifies that the future IPsec systems will use AES-CBC-128 for encryption and AES-XCBC-MAC-96 mode for authentication [**?**]. Figure 7 shows that these are also

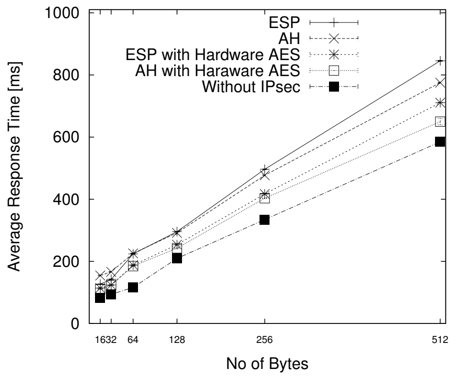
E. System-wide Energy Overhead

Securing the Internet of Things has a cost in terms of added energy usage. We measure the energy overhead of the available security options on the Tmote Sky using Contiki's integrated energy estimator. We measure the total number of CPU ticks from the reception of the first fragment of a message, when starting link layer decryption. We stop counting when the link layer encryption of the last packet is finished, but we ignore the network time between the packets. In total we the link layer processing, 6LoWPAN processing, μIP stack handling, and application-layer processing. These experiments are run with and without hardware support. For the

Figure 8 shows the energy consumption of Link Layer security only, IPsec using either AH or ESP, and without using any security. Since the variance of the 20 runs was very low, it is not 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 significantly higher than without IPsec we argue that this is negligible when compared to the consumption of typical radio chips. In the worst measured case, AH on 64 bytes, the energy consumed is around 0.5 mJ. The radio chip of the Tmote Sky consumes the same amount of energy after 8 ms of idle listening.

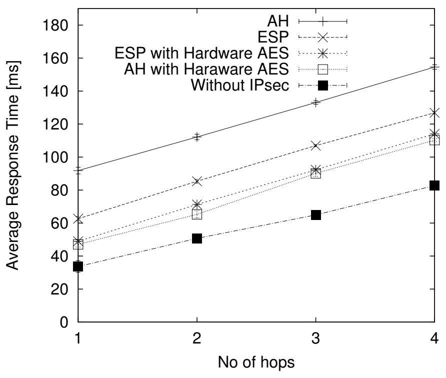


(a) Single Hop with different data sizes

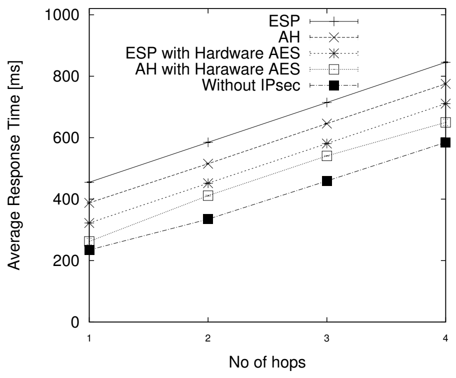


(b) Multi Hop (4) with different data sizes

Fig. 9: Response time versus datagram size with AH, ESP and without IPsec. ESP is faster than AH for small datagrams because it does not process the 40 bytes IP header. AH is faster than ESP for large datagrams because it processes authentication but no encryption.



(a) Multi Hop with 16 bytes data size



(b) Multi Hop with 512 byte data size

Fig. 10: Response time versus number of hops with AH, ESP and without IPsec. The overhead of IPsec is constant across a single hop and a multihop network.

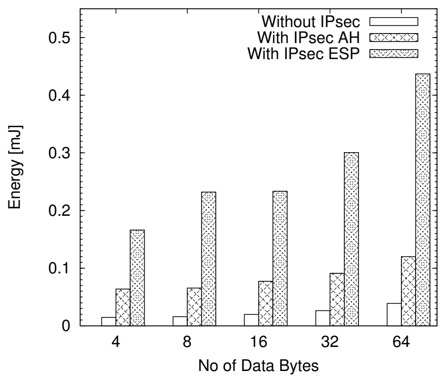


Fig. 8: Node energy consumption is lower without IPsec and higher for ESP than for AH. Compared to other activities e.g. idle listening it is not significant.

F. System-wide Response Time Overhead

We measure and evaluate the response time for different data sizes with IPsec and without IPsec. 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. We conduct experiments using a routing distance in the WSN ranging from 1 to 4 hops and for IP datagrams with a size ranging from 16 to 512 bytes. We execute every experiment 10 times.

Figure 9 shows the response time in dependency of the IP datagram size. When the datagram size is too large to fit a single 802.15.4 packet, the data are fragmented according to the 6LoWPAN standard. Consistently with the mirco-benchmarks in Figure 7, the overhead of IPsec grows linearly with datagram sizes. We observe that for small sizes, ESP is faster than AH. This is because unlike AH, ESP does not process the full 40 bytes IP header. With larger sizes, AH is faster than ESP, because it ensures authentication only, while ESP authenticates plus encrypts and decrypts the messages.

Figure 10 shows the response times obtained in dependence of hop distance. For a given data size, we observe that the overhead of either AH or ESP is constant, whatever the number of hops. This is because, for the intermediate nodes, the cost of forwarding the data with and without IPsec is the same; the overhead is only due to computation on the end nodes. In the worst case (512 bytes), we measured an overhead of 261 ms.

G. Improvements Using Hardware Support

The efficiency of IPSec can be improved by employing cryptographic functions provided by sensor node hardware. For example, the CC2420 radio chip present on many sensor node platforms provides such functionality. To investigate possible improvements we extend our prototype implementation to use this hardware for the required AES computations. Figure 9 and Figure 10 show the impact of hardware supported cryptography on the achievable response time. In all cases hardware-based implementations are faster than pure software implementations. When processing 512 byte datagrams over a single hop the overhead of pure software AH is 65 % which decreases to 12 % with the help of the cryptographic coprocessor. For ESP the decrease ranges from 64 % to 37 %.

VII. Conclusions and Future Work

WSNs will be an integral part of the Internet and IPv6 and 6LoWPAN are the protocol standards that are expected to be used in this context. IPsec is the standard method to secure Internet communication and we investigate if IPsec can be extended to sensor networks. Towards this end, we have presented the first IPsec specification and implementation for 6LoWPAN. We have extensively evaluated our implementation and demonstrated that it is possible and feasible to use compressed IPsec to secure communication between sensor nodes and hosts in the Internet.

To securely communicate with any IPv6 enabled node on the Internet pre-shared keys are sufficient but not very flexible. Therefore, we plan to investigate if an automatic key exchange protocol for 6LoWPANs based on IPsec's Internet Key Exchange protocol (IKE) is feasible.

Acknowledgments

This work has been performed within the SICS Center for Networked Systems funded by VINNOVA, SSF, KKS, ABB, Ericsson, Saab SDS, TeliaSonera, T2Data, Vendolocus and Peerialism. This work has been supported by VINNOVA, SSF and by the European Commission with contract FP7-2007-2-224053 (CONET). This work was also partially funded by ERCIM through the Alain Bensoussan postdoc fellowship program.