Implementing IPsec in Wireless Sensor Networks

Prabhakar Varadarajan
Department of Electrical and Computer Engineering
Southern Illinois University Carbondale
Carbondale, USA
prabhakar@siu.edu

Garth V. Crosby
Department of Technology
Southern Illinois University Carbondale
Carbondale, USA
garth.crosby@siu.edu

***Abstract*— There is an increasing need for wireless sensor networks (WSNs) to be more tightly integrated with the Internet. Several real world deployment of stand-alone wireless sensor networks exists. A number of solutions have been proposed to address the security threats in these WSNs. However, integrating WSNs with the Internet in such a way as to ensure a secure End-to-End (E2E) communication path between IPv6 enabled sensor networks and the Internet remains an open research issue. In this paper, the 6LoWPAN adaptation layer was extended to support both IPsec's Authentication Header (AH) and Encapsulation Security Payload (ESP). Thus, the communication endpoints in WSNs are able to communicate securely using encryption and authentication. The proposed AH and ESP compressed headers performance are evaluated via test-bed implementation in 6LoWPAN for IPv6 communications on IEEE 802.15.4 networks. The results confirm the possibility of implementing E2E security in IPv6 enabled WSNs to create a smooth transition between WSNs and the Internet. This can potentially play a big role in the emerging "Internet of Things" paradigm.**

Keywords—wireless sensor networks, 6LoWPAN, IPsec, AH, ESP

I. INTRODUCTION

Using IPv6 over Low Power Wireless Personal Area Networks (6LoWPAN) is a standard developed by a working group of the Internet Engineering Task Force (IETF). The work done by this group is published in two documents: RFC 4919 [1] and RFC 6282 [2]. The former describes the assumptions and goals for transmitting IP over IEEE 802.15.4 networks, while the later, RFC 6282, defines the compression format for IPv6 datagrams over IEEE 802.15.4 based networks. Real-world deployments of wireless sensor networks (WSNs) using IPv6 require a secure solution in the context of 6LoWPAN, which is currently an open issue. Although security solutions exist for particular scenarios, they do not provide a standard solution for WSNs.

There is a strong need to encrypt messages and the data flowing between nodes in WSNs to ensure message confidentiality. During communication, messages between IPv6 hosts are secured by default using IPsec. It is also possible to secure data flowing between IPv6 hosts and 6LoWPAN sensor nodes by using IPsec. In this paper, we evaluate the two new security extensions for IPv6: Authentication Header (AH) and Encapsulation Security Payload (ESP) that allow the protection of 6lowpan/IPv6 packets in WSNs.

The AH is used to provide data integrity and authentication while the ESP is used to provide data confidentiality, integrity and authentication. Both AH and ESP can be used to provide secure IPv6 packets. Depending on the nature of the applications AH or ESP can be used accordingly. The defined extension of IPsec [3,4] takes into consideration the interoperability between the Internet hosts and sensor nodes. It also states that IPsec security is already present on the IPv6-enabled hosts to which the nodes are communicating. 6LoWPAN uses the compression techniques that are already defined to reduce the size of the IPv6 header as well as extension header like the UDP transport header. For supporting IPsec's AH and ESP, additional IPv6 headers are included and care should be taken in compressing these extension headers to include them in each packet. As nodes are resource-constrained devices, compression must always be taken into consideration when defining new protocols and communications architectures for WSNs [5]. It is obvious that the support of IPSec in 6lowpan will increase packet size as additional headers are included. However, the usage of 802.15.4 link-layer security mechanisms can be completely removed, which in turn leaves some header space.

The main contribution of the paper is to present the results of our test-bed evaluation of an implementation of IPsec for a 6LoWPAN WSN connected to IPv6 internet hosts. We evaluated the implementation in terms of energy and code size in order to validate new security methods for WSNs. The remainder of the paper is as follows. Section II discusses related work regarding security at the network and higher layers. In section III, we describe the relevant protocols including IPv6, IPsec and 6LoWPAN and explain the usage of the proposed extension headers. Our results are presented in section IV. Lastly, section V concludes the paper.

II. RELATED WORK

Most of the security proposals for communication between wireless sensor nodes and Internet hosts are based on the modified versions of the Secure Sockets Layer (SSL) protocol. For example, SSNAIL [6] doesn't employ a secure gateway. Though SSNAIL provides end-to-end security between the Internet hosts and wireless sensor nodes using a lightweight version of the SSL protocol, it assumes that software for both entities is compatible when necessary modifications are made to the SSL protocol. Although, the security offered by SSL is good, it requires both sides of the endpoints to run SSL-aware applications during their communication session. The main limitation of SSL is that, it uses a standard configuration for security association and cryptographic algorithms, therefore creating unique solutions for specific WSN applications.

The message encryption and authentication algorithms

978-1-4799-3223-8/14/$31.00 ©2014 IEEE

used in WSN uses standard cryptographic mechanisms such as block ciphers in IEEE 802.15.4. However, it is difficult to implement them on the resource constrained wireless sensor nodes, because of the requirements of memory code space and processing speed. Additionally, there is no efficient way to implement the key distribution protocols on resource constrained devices. As a result, many researchers have attempted to reduce the resource requirements of the cryptographic mechanisms; for example, TinyEEC [7] and NanoEEC [8]. Others simplify the key distribution; for example, Liu *et al.* proposal for pairwise key predistribution [9] and DHB -KEY [10]. Although a number of improvements have been made with regards to cryptographic mechanisms, a standard way of implementing security on resource-constrained devices is still an open issue. IPsec has already been tested on some WSN products. For example, ArchRock PhyNET [11] implements IPsec in tunnel mode between the gateway and Internet hosts, but the security services offered completely depends on link-layer security, thereby breaching end-to-end security. We believe a lightweight version of IPsec integrated into 6LoWPAN can become a standardized way of providing end-to-end security.

A. Security for 6LoWPAN

The IEEE 802.15.4 [12] standard defines Advanced Encryption Standard (AES) message encryption and authentication for the link-layer. These algorithms can be implemented by the modification of hardware designed exclusively for the transceiver chip, but link-layer security provides only hop-by-hop security to the messages as they travel from one endpoint to another. However, end-to-end security at the network layer can be established using lightweight IPsec. Figure 1 illustrates how IPsec can be used to support 6LoWPAN to provide end-to-end security.

As per the specifications of 6LoWPAN mentioned in [1, 2] security support for IP - based low power wireless networks is still an open issue. Most of the security provided in 6LoWPAN, as of the writing of this paper, relies on link-layer security for providing data encryption and integrity. However, link layer security mechanism provides only hop-by-hop security. This implies that every node in the WSN has to be trusted. Another thing to note is that although the hardware of 802.15.4 supports link-layer mechanisms it doesn't provide true end-to-end security in its implementation. However, the IPsec protocol suite mandated by IPv6 provides true end-to-end security for IP communications [13].

Fig. 1: Use of IPsec to secure communication between sensor nodes and hosts in an IPv6 enabled internet. [4]

B. Background

For the convenience of the reader we describe the main functionalities of IPv6, IPsec and 6LoWPAN in the remainder of this section. For more information regarding these we refer you to the RFCs: RFC 2460[14], RFC 4301 [15] and RFC 6282 [2].

IPv6 is a new version of the internet protocol [14], which has an increased header size from 32 bits to 128 bits in order to allow trillions of devices to connect to the Internet. Hence, advancing the emergence of the 'Internet of Things' (IoT). Besides the increased address space provided by IPv6 compared to IPv4, it also has a simplified header format, better support for extension headers, and authentication and privacy capabilities.

IPsec [14] defines a set of protocols for the securing of IP based communication by providing authentication and privacy. It has two security protocols: Authentication header (AH) [15] and Encapsulation Security Payload (ESP) [16]. The IP security architecture uses Security Associations [15] for building security functions into IP. It is simply a bundle of algorithms for encryption and authentication including key exchange mechanisms for communication. Security associations are established using the Internet Key Exchange (IKE) protocol. It is implemented by pre-shared keys that work with any IPv6 enabled node on the Internet.

The main purpose of AH is to protect against replay attacks, provide data origin authentication and connectionless integrity that contains IP datagrams. The authenticated data is produced by the Message Authentication Code (MAC). The MAC is then applied to the IP header, AH header and IP payload. Figure 2, shows the basic AH which has a size of 24 bytes.

The function of ESP is to provide authentication, integrity and confidentiality for IP packets. The main difference between AH and ESP is that the former can encrypt the payload of IP, but AH doesn't secure the IP header. The payload of the IP is encrypted when the IP header is followed by ESP.

The ESP includes a SPI that contains an arbitrary value to identify the security association of the receiving party, a monotonically increasing sequence number to prevent replay attacks, the Encrypted payload, padding for encryption 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.

There are two different types of mode that are being supported by AH and ESP: Transport and Tunnel mode. In transport mode, only the payload of the IP is being encrypted and/or authenticated. In tunnel mode, a new IP header is placed in front of the original IP packet and the entire packet is authenticated and/or encrypted. Tunnel mode is usually used to create Virtual Private Networks (VPN). Since we are dealing with 6LoWPAN, tunnel mode does not seem to be a viable solution as it will further increase the packet size due to the additional headers. Hence, we focus on the transport mode.

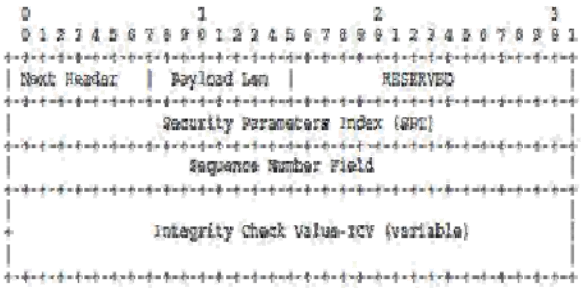


Fig. 2: AH header [15]

B. 6LoWPAN

The main goal of 6LoWPAN [2] is to integrate the existing IP based infrastructures and WSN by specifying how IPv6 packets can be transmitted over an IEEE 802.15.4 network by defining an adaptation layer and header compression mechanisms for IPv6 packets. In the IEEE 802.15.4 standard, the maximum physical layer size of the packet is 127 octets, which fits perfectly in the IPv6 packet. The maximum frame header size is 25 bytes and since the size of IEEE 802.15.4 packet is 127 bytes, an IPv6 packet therefore has to fit in 102 bytes. Given that the headers of the packet would already consume 48 bytes of the available 102 bytes; there is need for header compression mechanisms. HC13 [2] proposes context aware mechanisms for header compression: the LOWPAN_IPHC encoding for IPv6 header compression and the LOWPAN_NHC encoding for next header compression. Figure 3 shows IPHC.

Addresses are formed from the active message address, which allows the mote to generate an IP address from its link layer address. The length of the IPHC is 2 octets and 13 bits are already defined and used for header compression. The IPv6 header that is uncompressed also follows the IPHC encoding in the order they would appear in a normal IPv6 header [18].

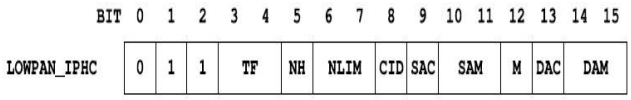
III. IPSEC AND 6LOWPAN

Though IPsec is also classified under extension headers, there has been no standard definition for IPsec so that packet sizes are reasonable in 6LoWPAN. One group of researchers Raza *et al.* [4] has defined encoding mechanisms for the AH and ESP.

A. IPv6 Extension Header Compression

The format of the LOWPAN_NHC octet is shown in Figure 4. The LOWPAN_NHC encodings for IPv6 extension headers are composed of a single LOWPAN_NHC octet followed by the IPv6 extension header. The defined NHC encoding was used to encode AH and ESP extension headers. The IPv6 extension header consists of NHC octet where three bits (bits 4, 5, 6) were used to encode the IPv6 extension header ID (EID). There are only eight possible values for EID and six are already defined by HC13 draft [2].

There are two remaining slots that are being reserved (101 and 110). These reserved slots were decided to encode AH and ESP. It is also necessary to set the NH field in LOWPAN_NHC to 1 to indicate that the next header AH and ESP is encoded using LOWPAN_NHC.



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: LOWPAN_IPHC base encoding [2]

B. Encoding for LOWPAN_NHC_AH

The first four bits in LOWPAN_NHC_AH represents the NHC ID for AH and it is set to 1101 as defined in [2]. These four bits are not necessary, but they are set to comply with the standards of 6LoWPAN. The EID field present in the previous NHC will identify whether the next header is AH or ESP. Figure 5 shows the structure of encoding for IPv6 Authentication Header.

- If PL = 0: Payload length (PL) is obtained from the value of Security Parameter Index (SPI) because the length of the authenticated data depends on the type of the algorithm that is being used and will be of a fixed size for any input. Therefore, the payload length present in AH is omitted.
  If PL = 1: The value of the payload is carried inline after the LOWPAN_NHC_AH header.
- If SPI = 0: The default value for the SPI has been set to 1. The default value for SPI is used by the sensor network and the SPI field is omitted. When no security association exists the value SPI 0 is reserved and it is used. This does not imply that all the nodes in a WSN use the same Security Association (SA), but that every node has a single preferred SA, identified by SPI 1.
  If SPI = 0: SPI is 32 bits, all the 33 bits indicating the SPI will be carried inline after the LOWPAN_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 the 32 bits of the sequence number will be carried inline after the LOWPAN_NHC_AH header.
- If NH = 0: The next header that is present in the AH will be used to specify the next layer that may be ESP or upper layer header and it is carried inline.

If NH = 1: The next header field present in the AH is omitted. The next header will be encoded using LOWPAN_NHC_AH.

C. Encoding for LOWPAN_NHC_ESP

The first four bits in LOWPAN_NHC_ESP represents the NHC ID for ESP and is set to 1110 as defined in [2]. These four bits are not necessary, but they are set just to comply with the standards of 6LoWPAN. The EID field present in the previous NHC will identify whether the next header is ESP or upper layer headers. Figure 6 shows the structure of encoding for IPv6 Encapsulation Security Payload.

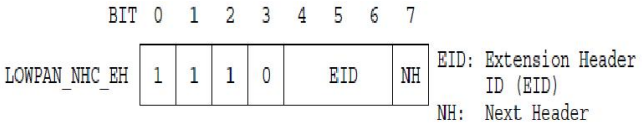


Fig. 4: IPv6 Extension Header Encoding

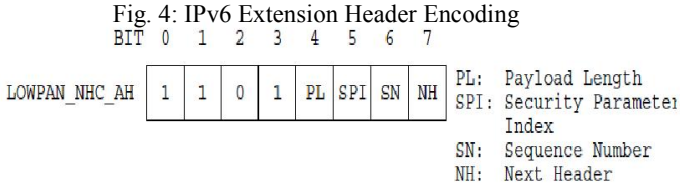


Fig. 5: LOWPAN_NHC_AH Encoding

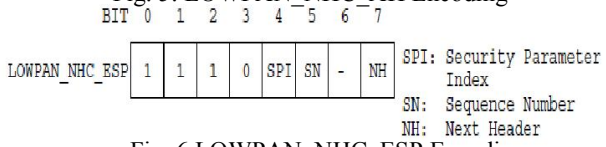


Fig. 6 LOWPAN_NHC_ESP Encoding

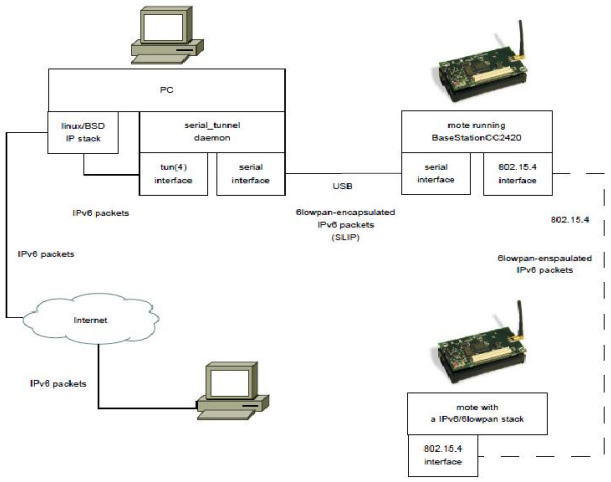


Fig. 7: Tunneling Daemon and the Internet

D. 6LoWPAN for Linux

A tunneling daemon was already developed to exchange packets between the motes and the PC [17]. The scenario is shown in Figure 7. Most of the PCs today do not have an implementation of 6LoWPAN or 802.15.4 on Linux operating systems. Thus, a tunneling daemon has been developed to use a mote as an 802.15.4 interface for a Linux PC. The mote is connected to the PC via the USB interface and runs the Base station application from TinyOS for forwarding the traffic between the 802.15.4 and the USB interface of the mote.

The translating daemon on the PC is a C program exchanging packets between the USB interface and a tun interface. The tun interface is a virtual network allowing a user space process to read and write packets to it. The daemon decapsulates the 6LoWPAN-encapsulated IPv6 packets coming from the mote and encapsulates the IPv6 packets on the tun interface. Thus, allowing standard IPv6 applications on Linux for communication with the motes without modifying the Linux kernel. Additionally, by enabling IPv6 forwarding on the PC, the motes can be connected to the Internet [17].

IV. EVALUATION AND RESULTS

A. Implementation and Experimental Setup

The implementation was tested using a scenario shown in Figure 7. The Implementation of IPsec AH and ESP are done on TinyOS 2.1.2 [18] using TelosB motes.

The implementation required the modification of the 6LoWPAN stack that is already implemented in TinyOS, providing 6LoWPAN functionality. The daemon discussed above is modified to include IPSec on PC and the 6LoWPAN application was adapted in order to provide the IPsec compression mechanism discussed in section III. The modified 6LoWPAN stack provides LOWPAN_NHC_EH, LOWPAN_NHC_AH and LOWPAN_NHC_ESP. The stack is then implemented on the sensor motes, using the tunneling daemon for connecting the WSN and the Internet.

We used the SHA1 and AES implementations from MIRACL [19], an open source library, and implement all cryptographic modes of operation needed for authentication and encryption in IPsec. For AH we implemented the mandatory AES-XCBC-MAC-96. For ESP we used the mandatory AES-CTR for encryption and AES-XCBC-MAC-96 for authentication. The proposed standard for IPsec by Cryptographic Suites [20] specifies that the future IPsec systems will use AES-CBC-128 for encryption and AES-XCBC-MAC-96 mode for authentication.

At this moment, TinyOS IPsec implementation doesn't support key exchange mechanism such as the Internet Key Exchange (IKE) protocol. Keys must be set manually before deployment. Moreover, it is to be noted that manual key distribution is used currently for traditional IEEE 802.15.4 link layer security.

B. Memory Footprint

We measured the ROM and RAM footprints of the TelosB mote for the IPsec implementation. Table 5.1 compares IPsec AH and ESP using the AES-CTR and AES-XCBX-MAC-96 modes of operation. The footprints are compared with a reference TinyOS system including 6LoWPAN. The ROM footprint overhead ranges from 0.1 kB (AH with XCBC-MAC-96) to 0.3 kB (ESP with (AES-CTR + AES-XCBC-MAC-96). This always keeps the system footprint under 16K bytes, the Flash ROM size of the Telosb mote. It is also beneficial to use AES-CTR + AES-XCBC-MAC-96 configuration without the AES decryption, resulting in a low memory footprint.

The RAM footprint of the TelosB mote is calculated as

the sum of the runtime stack usage and global data at the time of compilation. With an additional footprint of 0.8kB, the RAM usage lies between 0.3 and 0.8 kB. These results show that both IPsec AH and ESP can be embedded in resource constrained devices while leaving space for applications.

TABLE I. ROM and RAM Footprints

	ROM footprint (kB)		**RAM footprint (kB)**	
System	***overall***	***overhead***	***overall***	***additional***
Without IPsec	14.2	-	2.2	-
With IPsec	14.6	0.4	3.0	0.8
AH with XCBC-MAC-96	14.3	0.1	2.5	0.3
ESP with AES-CTR + AES-XCBC-MAC-96	14.5	0.3	2.5	0.3

Note that Table I ROM and RAM footprints show that AH and ESP consumes just 0.1kB and 0.3kB

V. CONCLUSION

In this paper, we implemented and evaluated the usage of the new compressed IPsec headers, together with cryptographic algorithms typically used with IP security architectures. The results presented demonstrate that it is feasible to utilize IPsec to provide end-to-end security in the emerging 'Internet of Things'. For future work we hope to investigate dynamic key distribution for the motes in the WSN-Internet communication network.

REFERENCES

[1] N. Kushalnagar , G. Montenegro and C.Schumacher "RFC 4919 - IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals," Aug 2007. Accessed September 2013 at http://tools.ietf.org/html/rfc4919

[2] J. Hui, Ed. and P.Thubert "RFC 6282 – Compression Format for IPv6 Datagrams over IEEE 802.15.4 – Based Networks", September 2011. Accessed September 2013 at http://tools.ietf.org/search/rfc6282

[3] S.Park, K.Kim,W.Haddad, S.Chakrabathi and J.Laganier "IPv6 over low power WPAN security analysis," March 15, 2011. Accessed September 2013 at http://tools.ietf.org/html/draft-daniel-6lowpan-security-analysis-05

[4] Shahid Raza, Simon Duquennoy, Tony Chung, Dogan Yazar, Thiemo Voigt and Utz Roedig " Securing Communication in 6LoWPAN with compressed IPsec" in the proceedings of the International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS), Barcelona, Spain, June 27-29, 2011.

[5] John Paul Walters, Zhengqiang Liang,Weisong Shi, and Vipin Chaudhary "Wireless Sensor Network Security: A Survey", in Yang Xiao (Eds), Security in Distributed, Grid, and Pervasive Computing, Auerbach Publications, CRC Press, 2006, pp. 1-50.

[6] Wooyoung Jung, Sungmin Hong, Minkeun Ha, Young-Joo Kim, Daeyoung Kim, "SSL-based Lightweight Security of IP-based Wireless Sensor Networks", in the proceedings of the International Conference on Advanced Information Networking and Application Workshop (WAINA), Bradford, United Kingdom, May 26-29, 2009

[7] A. Liu and P. Ning, "TinyECC: A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks", in the proceedings of the International Conference on Information Processing in Sensor Networks (*IPSN),* St. Louis, Missouri, USA, April 22-24, 2008.

[8] P. Szczechowiak, L. Oliveira, M. Scott, M. Collier, and R. Dahab, "NanoECC: Testing the limits of elliptic curve cryptography in sensor networks" in the proceedings of the 5th European Conference on Wireless Sensor Networks (*EWSN)*, in Roberto Verdone (Eds), Lecture Notes in Computer Science (LNCS), Springer-Verlag Berlin, Heidelberg, January 30, 2008. pp. 305-320

[9] D. Liu and P. Ning, "Establishing Pairwise Keys in Distributed Sensor Networks", in the proceedings of the *10th ACM Conference on Computer and Communications Security (CCS)*, New York City, NY, USA, 2003.

[10] T. Chung and U. Roedig, "DHB-KEY: An Efficient Key Distribution Scheme for Wireless Sensor Networks", in the proceedings of the 5th IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS), Atlanta, Georgia, USA, September 29- October 2, 2008.

[11] ArchRock Corporation. Phynet n4x series, 2008. Accessed on September 16, 2013 at http://www.businesswire.com/news/home/20081014005655/en/Arch-Rock-Adds-Ruggedized-Version-IP-Based-PhyNet#.Uve51hb12Cs

[12] IEEE Computer Society. IEEE std. 802.15.4-2006, 2006.

[13] S. Kent and R. Atkinson, "Security Architecture for the Internet protocol", RFC 2401, 1998. Accessed on September 16, 2013 at http://www.ietf.org/rfc/rfc2401.txt

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

[15] Stephen Kent. IP Authentication Header. RFC 4302, 2005.

[16] S. Kent. IP Encapsulating Security Payload. RFC 4303, 2005.

[17] M. Harvan, J. Schönwälder, "A 6lowpan Implementation for TinyOS 2.0", RWTH Aachen, Department of Computer Science, Technical Report, July 2007. Acessed on September 16 at http://aib.informatik.rwth-aachen.de/

[18] http://www.tinyos.net/

[19] Shamus Software. Multiprecision Integer and Rational Arithmetic C/C++ Library, http://www.compapp.dcu.ie/~mike/shamus.html

[20] P. Ho Man. Cryptographic Suites for IPsec. RFC 4308, December 2005.