Mälardalen University Doctoral Thesis
No.139

Lightweight Security Solutions for the Internet of Things

Shahid Raza

June 2013

Department of Computer Science and Engineering
Mälardalen University
Västerås, Sweden

ISSN 1651-4238
ISBN 978-91-7485-110-6
Printed by Mälardalen University, Västerås, Sweden
Distribution: Mälardalen University Press

Swedish Institute of Computer Science
Doctoral Thesis
SICS Dissertation Series 64

Lightweight Security Solutions for the Internet of Things

Shahid Raza

2013

Swedish Institute of Computer Science(SICS)
SICS Swedish ICT, Kista
Stockholm, Sweden

ISSN 1101-1335
ISRN SICS-D–64–SE
Printed by Mälardalen University, Västerås, Sweden

Abstract

The future Internet will be an IPv6 network interconnecting traditional computers and a large number of smart objects or networks such as Wireless Sensor Networks (WSNs). This Internet of Things (IoT) will be the foundation of many services and our daily life will depend on its availability and reliable operations. Therefore, among many other issues, the challenge of implementing secure communication in the IoT must be addressed. The traditional Internet has established and tested ways of securing networks. The IoT is a hybrid network of the Internet and resource-constrained networks, and it is therefore reasonable to explore the options of using security mechanisms standardized for the Internet in the IoT.

The IoT requires multi-faceted security solutions where the communication is secured with confidentiality, integrity, and authentication services; the network is protected against intrusions and disruptions; and the data inside a sensor node is stored in an encrypted form. Using standardized mechanisms, communication in the IoT can be secured at different layers: at the link layer with IEEE 802.15.4 security, at the network layer with IP security (IPsec), and at the transport layer with Datagram Transport Layer Security (DTLS). Even when the IoT is secured with encryption and authentication, sensor nodes are exposed to wireless attacks both from inside the WSN and from the Internet. Hence an Intrusion Detection System (IDS) and firewalls are needed. Since the nodes inside WSNs can be captured and cloned, protection of stored data is also important.

This thesis has three main contributions. (i) It enables secure communication in the IoT using lightweight compressed yet standard compliant IPsec, DTLS, and IEEE 802.15.4 link layer security; and it discusses the pros and cons of each of these solutions. The proposed security solutions are implemented and evaluated in an IoT setup on real hardware. (ii) This thesis also presents the design, implementation, and evaluation of a novel IDS for the IoT. (iii) Last but

not least, it also provides mechanisms to protect data inside constrained nodes. The experimental evaluation of the different solutions shows that the resource-constrained devices in the IoT can be secured with IPsec, DTLS, and 802.15.4 security; can be efficiently protected against intrusions; and the proposed combined secure storage and communication mechanisms can significantly reduce the security-related operations and energy consumption.

Sammanfattning

Framtidens Internet är ett IPv6-nätverk vilket förbinder traditionella datorer och ett stort antal smarta objekt eller nätverk som trådlösa sensornätverk (WSN). Detta Internet of Things (IoT) kommer att vara grunden för många tjänster och vårt dagliga liv kommer att bero pådess tillgänglighet och säkra drift. Därför måste man bland många andra frågor adressera utmaningen att skapa säker kommunikation i Internet of Things. Det traditionella Internet har etablerat och testat olika sätt att skapa säkra nätverk. IoT är en blandning av nätverk, av Internet och nät med småresurser, och det är därför viktigt att undersöka möjligheterna att använda säkerhetsmekanismer standardiserade för Internet i Internet of Things.

Internet of Things kräver mångfacetterade säkerhetslösningar där kommunikationen är säkrad med sekretess, integritet och autentisering av tjänster, nätverket skyddas mot intrång och störningar, och data inuti en sensornod lagras i krypterad form. Med standardiserade mekanismer kan kommunikationen säkras i olika skikt: i länkskiktet med IEEE 802.15.4-säkerhet, i nätskiktet med IP-säkerhet (IPsec), och i transportskiktet med Datagram Transport Layer Security (DTLS) . ven när kommunikationen är säkrad med kryptering och autentisering är sensornoderna utsatta både för trådlösa attacker inifrån WSN och från Internet. Därför behövs ett system för att upptäcka intrång (Intrusion Detection System, IDS), och även brandväggar behövs. Eftersom noderna inne i WSN kan stjälas och klonas, är skyddet av lagrade data ocksåviktigt.

Denna avhandling har tre huvudsakliga bidrag. (i) Den möjliggör säker kommunikation i Internet of Things med lättviktiga, komprimerade, men standardkompatibla IPsec, DTLS och IEEE 802.15.4-länkskiktssäkerhet, och jämför för- och nackdelar mellan dessa lösningar. De föreslagna säkerhetslösningarna implementeras och utvärderas i en IoT-installation påriktig hårdvara. (ii) Denna avhandling presenterar ocksådesign, implementation och utvärdering av ett nytt IDS för Internet of Things. (iii) Sist men inte minst, avhandlingen pre-

senterar ocksåmekanismer för att skydda data i noder med begränsade resurser. Den kvantitativa utvärderingen av de olika lösningarna visar att enheter i IoT med begränsade resurser kan säkras med IPsec, DTLS och 802.15.4-säkerhet, och kan effektivt skyddas mot intrång, och den föreslagna kombinationen av säker lagring och mekanismer för säker kommunikation kan avsevärt minska kostanden för säkerhetsrelaterade operationer och energiförbrukning.

Acknowledgements

First and foremost, I am thankful to Almighty Allah for bestowing me health, persistence, and knowledge to complete this work. I implore Him to make my knowledge and skills useful to mankind.

I am obliged to all the people in SICS Swedish ICT, Mälardalen University, and ABB who were associated with this work and guided me throughout the thesis period, but it is worth mentioning some of the people who were really benevolent and supportive. I first express my gratitude to my advisor Prof. Thiemo Voigt for his unprecedented support, extensive guidance, and personal involvement in all phases of this research. Without his encouragement, guidance, and keen interest this thesis would not have been completed.

I am deeply indebted and grateful to my supervisors Prof. Mats Björkman, Dr. Christian Gehrmann, Prof. Seif Haridi, and Thiemo Voigt for providing me the much needed motivation, inspiration and guidance in achieving this milestone. Its been pleasure to work with the co-authors around the globe. I genuinely thank Utz Roedig, Ibrahim Ethem Bagci, and Tony Chung from Lancaster University; Krister Landernäs and Mikael Gidlund for ABB; Gianluca Dini from University of Pisa; Kasun from Uppsala University; René Hummen from RWTH Aachen University; and Adriaan, Dogan, Hossein, Joel, Linus, Simon, and Thiemo from SICS.

I am very grateful to Dr. Sverker Janson, head of the Computer Systems Laboratory (CSL) and a supportive mentor, for helping me in all academic and non-academic matters whenever needed. I am thankful to my current and formers co-workers in NES group: Adriaan, Adam, Beshr, Dogan, Fredrik, Joakim, Joel, Luca, Niclas, Nicolas, Niklas, Prasant, Zhitao, and obviously Simon and Thiemo. I acknowledge all colleagues at SICS particularly Mudassar Aslam, Eva Gudmundsson, Jerker Berg, Thomas Ringström, Lotta Jörsäter, Karin Fohlstedt, Vicki Knopf, Bengt Ahlgren, Maria Holm, Oliver Schwarz, Orc Lönn, Rolf Blom, and of course Janusz Launberg and Christer Norström.

Last, but certainly not least, I cannot thank my family enough for the unend-

ing affection, encouragement, respect and all the exciting and gloomy things I have shared with them. I express my deepest gratitude to my parents, brothers, sisters, my wife, and my son for their emotional and moral support throughout my academic career and also for their tolerance, inspiration, and prayers.

Shahid Raza
Stockholm, May, 2013

This work has been performed in the Networked Embedded Systems (NES) Group that is a part of the Computer Systems Laboratory in the SICS Swedish ICT. This work is mainly financed by the Higher Education Commission (HEC) Pakistan in the form of PhD scholarship, and SICS Center for Networked Systems (CNS). The SICS CNS is funded by VINNOVA, SSF, KKS, ABB, Ericsson, Saab SDS, TeliaSonera, T2Data, Vendolocus, and Peerialism. This work has been partially supported by SSF, Uppsala VINN Excellence Center for Wireless Sensor Networks (WISENET), and European Commission with contract FP7-2007-2-224053 (CONET), FP7-2007-2-224282 (GINSENG), and FP7-ICT-2011.1.3- 288879 (CALIPSO).
The SICS Swedish ICT is sponsored by TeliaSonera, Ericsson, Saab SDS, FMV (Defence Materiel Administration), Green Cargo (Swedish freight railway operator), ABB, and Bombardier Transportation.

List of publications

Publications included in the thesis

1. Shahid Raza, Adriaan Slabbert, Thiemo Voigt, Krister Landernäs. Security Considerations for the WirelessHART Protocol. *In proceedings of 14th IEEE International Conference on Emerging Technologies and Factory Automation* (ETFA'09), September 22-26, 2009, Mallorca, Spain.

2. Shahid Raza, Simon Duquennoy, Tony Chung, Dogan Yazar, Thiemo Voigt, Utz Roedig. Securing Communication in 6LoWPAN with Compressed IPsec. *In proceedings 7th IEEE International Conference on Distributed Computing in Sensor Systems* (DCOSS '11), June 27-29 2011, Barcelona, Spain.

3. Shahid Raza, Simon Duquennoy, Joel Hoglund, Utz Roedig, Thiemo Voigt. Secure Communication for the Internet of Things - A Comparison of Link-Layer Security and IPsec for 6LoWPAN. *Journal of Security and Communication Networks*, Early View (), Wiley, 2012.

4. Shahid Raza, Hossein Shafagh, Kasun Hewage, René Hummen, Thiemo Voigt. Lithe: Lightweight Secure CoAP for the Internet of Things. [In Submission]

5. Shahid Raza, Linus Wallgren, Thiemo Voigt. SVELTE: Real-time Intrusion Detection in the Internet of Things. *Ad Hoc Networks* Journal, Elsevier, 2013. [Accepted]

6. Ibrahim Ethem Bagci, Shahid Raza, Tony Chung, Utz Roedig, Thiemo Voigt. Combined Secure Storage and Communication for the Internet of Things. *In proceedings of 10th IEEE International Conference on Sensing, Communication, and Networking* (SECON'13), June 24-27, 2013, New Orleans, USA.

Other publications
In addition to the papers included in the thesis I have also co-authored the following papers:

1. René Hummen, Jan H. Ziegeldorf, Hossein Shafagh, Shahid Raza, Klaus Wehrle. Towards Viable Certificate-based Authentication for the Web of Things. In proceedings of *ACM Workshop on Hot Topics on Wireless Network Security and Privacy*, co-located with ACM WiSec 2013, April 17-19, 2013, Budapest, Hungary.

2. Daniele Trabalza, Shahid Raza, Thiemo Voigt. INDIGO: Secure CoAP for Smartphones- Enabling E2E Secure Communication in the 6IoT. In proceedings of *International Conference on Wireless Sensor Networks for Developing Countries* (WSN4DC 13), April 24-26 2013, Jamshoro, Pakistan.

3. Ibrahim E. Bagci, Mohammad R. Pourmirza, Shahid Raza, Utz Roedig, Thiemo Voigt. Codo: Confidential Data Storage for Wireless Sensor Networks. In proceedings of *8th IEEE International Workshop on Wireless and Sensor Networks Security* (WSN'S 2012), in conjunction with 9th IEEE MASS'2012, October 8-12 2012, Las Vegas, Nevada, USA.

4. Shahid Raza, Daniele Trabalza, Thiemo Voigt. Poster Abstract: 6LoWPAN Compressed DTLS for CoAP. In proceedings of *8th IEEE International Conference on Distributed Computing in Sensor Systems* (DCOSS '12), 16-18 May 2012, Hangzhou, China.

5. Shahid Raza, Thiemo Voigt, Vilhelm Juvik. Lightweight IKEv2: A Key Management Solution for both Compressed IPsec and IEEE 802.15.4 Security. In *IETF Workshop on Smart Objects Security*, March 23, 2012, Paris, France.

6. Shahid Raza, Simon Duquennoy, Tony Chung, Dogan Yazar, Thiemo Voigt, Utz Roedig. Demo Abstract: Securing Communication in 6LoWPAN with Compressed IPsec. In proceedings *7th IEEE International*

Conference on Distributed Computing in Sensor Systems (DCOSS '11), 27-29 June 2011, Barcelona, Spain.

7. Shahid Raza, Gianluca Dini, Thiemo Voigt, and Mikael Gidlund. Secure Key Renewal in WirelessHART. In *Real-time Wireless for Industrial Applications* (RealWin'11), CPS Week, 11-16 April 2011, Chicago, Illinois, USA.

8. Shahid Raza, Thiemo Voigt, and Utz Roedig. 6LoWPAN Extension for IPsec. In *Interconnecting Smart Objects with the Internet Workshop*, 25 March 2011, Prague, Czech Republic.

9. Auriba Raza and Iftikhar A, Raja and Elisabet Lindgren and Shahid Raza. Land-use Change Analysis of District Abbottabad Pakistan: Taking Advantage of GIS and Remote Sensing. In proceedings of *4th International conference on Environmentally Sustainable Development*, June 2011, Pakistan.

10. Shahid Raza and Thiemo Voigt. Interconnecting WirelessHART and Legacy HART Networks. In proceedings of *1st International Workshop on Interconnecting Wireless Sensor Network* in conjunction with DCOSS'10., 21-23 June 2010, UC Santa Barbara, USA.

11. Shahid Raza, Thiemo Voigt, Adriaan Slabbert, Krister Landernäs. Design and Implementation of a Security Manager for WirelessHART Networks. In proceedings of *5th IEEE International Workshop on Wireless and Sensor Networks Security* (WSN'S 2009), in conjunction with MASS'2009, 12-15 Oct 2009, Macau SAR, P.R.C..

12. Joakim Eriksson, Fredrik Österlind, Thiemo Voigt, Niclas Finne, Shahid Raza, Nicolas Tsiftes, and Adam Dunkels. Demo abstract: accurate power profiling of sensornets with the COOJA/MSPSim simulator. In proceedings of *6th IEEE International Conference on Mobile Ad-hoc and Sensor Systems* (IEEE MASS 2009), 12-15 Oct 2009, Macau SAR, P.R.C..

Contents

I

Thesis

Chapter 1

Introduction

The Internet of Things (IoT) is a network of globally identifiable physical objects (or things), their integration with the Internet, and their representation in the virtual or digital world. In order to build the IoT, a wide range of technologies are involved. For example, RFID for location and device identification, improved personal and wide area networking protocols, web technologies, etc. These technologies help to build a virtual world of things on top of the physical world where things through Machine-to-Machine (M2M) communication talk to each other, through humans-to-machine interactions provide information to humans or take actions on human inputs, or act as passive entities to provide data to intelligent entities. Wireless Sensor Networks (WSN) is one such technology that connects the virtual world and the physical world where nodes can autonomously communicate among each other and with intelligent systems. This thesis focuses on the IoT formed through the interconnection of IP-connected WSNs and the Internet.

A conventional WSN is a network of sensor devices that sense and collect environmental data and cooperatively forward it to the sink node for further processing. These first generation WSNs lack any standardization support, are mostly used for environmental monitoring, and are deployed in remote areas such as forests, deserts, volcanos, and battlefields. Current WSNs are deployed in environments more close to humans and aimed for applications such as building automation, bridge and tunnel monitoring, industrial automation and control, and human sensing. The sink in current WSNs, such as WirelessHART networks, can query data from sensor nodes and/or send control messages to them. Though some standards are being developed for industrial WSNs such

as WirelessHART and ISA100.11a, there exists no specific standards for routing, addressing, security, etc. for such networks. Therefore, building current WSNs requires specialized skills in software and hardware development and protocol design. Also, conventional WSNs are not interoperable, require complex gateways, and are not scalable.

Sensor nodes are resource-constrained devices with limited storage and processing capabilities, are battery powered, and are connected through lossy links. The Internet Protocol (IP) is also proposed for WSN [1]; until recently IP has been assumed to be too heavyweight protocol to be used in WSN, as additional 40 bytes of IPv6 header are added in each packet [2]. However, IP offers interoperability, scalability, easy of programing, has ready to use hardware, eliminates the need of complex gateways, and has pool of readily available experts. Considering these advantages, IPv6 over low-powered Personal Area Network (6LoWPAN) [3, 4] is standardized. With the advent of 6LoWPAN, it is possible to use IP in resource-constrained WSNs in an efficient way [5]; such networks are called 6LoWPAN networks.

1.1 The IPv6-connected Internet of Things

With the introduction of 6LoWPAN compressed IPv6 in WSNs, resource constrained devices can be connected to the Internet. This hybrid network of the Internet and the IPv6 connected constrained devices form the IoT. Unlike the Internet where devices are mostly powerful and unlike typical WSN where devices are mostly resource constrained, the things in the IoT are extremely heterogeneous. An IoT device can be a typical sensor node, a light bulb, a microwave oven, an electricity meter, an automobile part, a smartphone, a PC or a laptop, a powerful server machine or even a cloud. Hence the number of potential devices that can be connected to the IoT are in hundreds of billions. This requires the use of IPv6 [16], a new version of the Internet Protocol that increases the address size from 32 bits to 128 bits (2^{128} unique addresses). Also, a number of protocols are being standardized to fulfill the specific needs of the IoT.

This section highlights the novel IoT technologies; Section 1.2 specifies the security requirements for the IoT that is developed based on these technologies; and Chapter 2 highlights challenges in providing secure communication in the IoT, and summarizes the contribution of this thesis towards securing the IoT.

6LoWPAN 6LoWPAN integrates IP-based infrastructures and WSNs by specifying how IPv6 packets are to be routed in constrained networks such as IEEE 802.15.4 networks [6]. To achieve this, the 6LoWPAN standard proposes context aware header compression mechanisms: the IP Header Compression (IPHC) for the IPv6 header, and Next Header Compression (NHC) for the IPv6 extension headers and the User Datagram Protocol (UDP) header. Due to the limited payload size of the link layer in 6LoWPAN networks, the 6LoWPAN standard also defines fragmentation and reassembly of datagram. 6LoWPAN defines a fragmentation scheme in which every fragment contains a reassembly tag and an offset. When security is enabled or for big application data size, the IEEE 802.15.4 frame size may exceed the Maximum Transmission Unit (MTU) size of 127 bytes; in that case additional fragment(s) are needed.

In order to allow compression of header like structures in the UDP payload and the layers above, an extension to the 6LoWPAN header compression, called Generic Header Compression (GHC) is also defined [7]. 6LoWPAN networks are connected to the Internet through the 6LoWPAN Border Router (6BR) that is analogous to a sink in a WSN. The 6BR preforms compression/decompression and fragmentation/assembly of IPv6 datagrams.

CoAP Due to the low-powered and lossy nature of wireless networks in the IoT, connection-less UDP, instead of stream-oriented TCP, is mostly used in the IoT. The synchronous Hyper Text Transfer Protocol (HTTP) is designed for TCP and is infeasible to use in the UDP-based IoT. Therefore, the Constrained Application Protocol (CoAP) [8], a subset of HTTP is being standardized as a web protocol for the IoT. CoAP is tailored for constrained devices and for machine-to-machine communication.

RPL Routing in constrained networks in the IoT, with limited energy and channel capacity, is achieved using the recently standardized the IPv6 Routing Protocol for Low-power and Lossy Networks (RPL) [9]. The RPL protocol creates a Destination-Oriented Directed Acyclic Graph (DODAG) that aims to prune path cost to the DAG root. RPL supports both uni-directional traffic to a DODAG root (typically the 6BR) and bi-directional traffic between constrained nodes and a DODAG root. Each node in the DODAG has a node ID (an IPv6 address), one or more parents (except for the DODAG root), and a list of neighbors. Nodes have a rank that determines their location relative to the neighbors and with respect to the DODAG root. The rank should always increase from the DODAG root towards nodes. In-network routing tables are maintained to

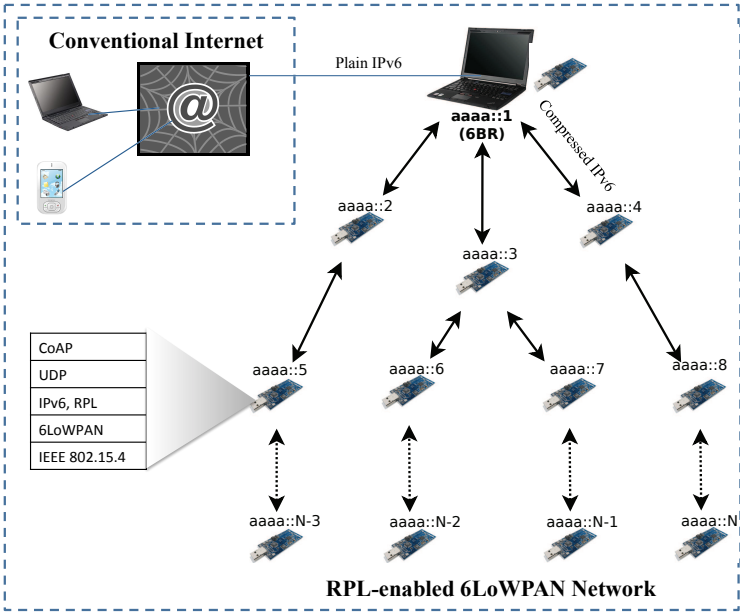


Figure 1.1: An interconnection of the Internet and WSNs using the novel IoT technologies 6LoWPAN, CoAP, and RPL which provide IPv6 support, web capabilities, and routing, respectively.

separate packets heading upwards and packets heading downwards in the network; this is called *storing* mode. RPL also supports *non-storing* mode where intermediate nodes do not store any routes.

Figure 1.1 shows an IoT setup that is build upon the novel technologies discussed in this section; the focus of this thesis is to protect this IoT with standard-based solutions.

1.2 Secure Internet of Things

IPv6 offers interconnection of almost every physical object with the Internet. This leads to tremendous possibilities to develop new applications for the IoT, such as home automation and home security management, smart energy moni-

toring and management, item and shipment tracking, surveillance and military, smart cities, health monitoring, logistics monitoring and management. Due to the global connectivity and sensitivity of applications, security in real deployments in the IoT is a requirement [10, 11]. The following security services [12] are necessary in the IoT.

Confidentiality: Messages that flow between a source and a destination could be easily intercepted by an attacker and secret contents are revealed. Therefor, these messages should be hidden from the intermediate entities; in other words, End-to-End (E2E) message secrecy is required in the IoT. Also, the stored data inside an IoT device should be hidden from unauthorized entities. *Confidentiality* services ensure this through encryption/decryption.

Data Integrity: No intermediary between a source and a destination should be able to undetectably change secret contents of messages, for example a medical data of a patient. Also, stored data should not be undetectably modified. Message Integrity Codes (MIC) are mostly used to provide this service.

Source Integrity or Authentication: Communicating end points should be able to verify the identities of each other to ensure that they are communicating with the entities who they claim to be. Different authentication schemes exist [13].

Availability: For smooth working of the IoT and access to data whenever needed, it is also important that services that applications offer should be always available and work properly. In other words, intrusions and malicious activities should be detected. Intrusion Detection Systems (IDSs) and firewalls, in addition to the security mechanisms above, are used to ensure availability security services.

Replay Protection: Last but not least, a compromised intermediate node can store a data packet and replay it at later stage. The replayed packet can contain a typical sensor reading (e.g. a temperature reading) or a paid service request. It is therefore important that there should be mechanisms to detect duplicate or replayed messages. Replay protection or freshness security services provide this, which can be achieved through integrity-protected timestamps, sequence numbers, nonces, etc.

In order to provide multi-faceted security, we need to ensure E2E communication security in the IoT, network security in 6LoWPAN networks, and also data-at-rest security to protect stored secrets and data.

1.2.1 Communication Security

Communication in the IoT should be protected by providing the security services discussed above. Using standardized Internet security mechanisms we

can provide communication security at different layers of the IP stack; each solution has its own pros and cons. Broadly speaking, the communication security can be provided E2E between source and destination, or on a per-hop basis between two neighboring devices. Table 1.1 shows an IoT stack with standardized security solution at different layers.

Link Layer: IEEE 802.15.4 Security

6LoWPAN networks use the IEEE 802.15.4 protocol [6] as link layer. 802.15.4 link-layer security [14] is the current state- of-the-art security solution for the IoT. The link layer security protects a communication on a per-hop base where every node in the communication path has to be trusted. A single pre-shared key is used to protect all communication. In case an attacker compromises one device it gains access to the key, and the security of the whole network is compromised. Per-hop security can detect the message modification on each hop unlike E2E where modified packets traverse the entire path up to the destination to be detected. Per-hop security with at least integrity protection should be used in 6LoWPAN networks to prevent unauthorized access through the radio medium, and to defend against effortless attacks launched to waste constrained resources. Though link-layer security is limited to securing the communication link between two neighboring devices, it is a flexible option and it can operate with multiple protocols at the layers above. For example with link-layer security enabled we can run both IP and non-IP protocols at the network layer.

Network Layer: IP Security

In the Internet and hence in the IoT, security at the network layer is provided by the IP Security (IPsec) protocol suite [15, 16, 17]. IPsec in transport mode provides end-to-end security with authentication and replay protection services in addition to confidentiality and integrity. By operating at the network layer, IPsec can be used with any transport layer protocol including TCP, UDP, HTTP, and CoAP. IPsec ensures the confidentiality and integrity of the IP payload using the Encapsulated Security Payload (ESP) protocol [17], and integrity of the IP header plus payload using the Authentication Header (AH) protocol [16]. IPsec is mandatory in the IPv6 protocol [2, 18] meaning that all IPv6 ready devices by default have IPsec support, which may be enabled at any time. Being a network layer solution, IPsec security services are shared among all applications running on a particular machine. However, being mandatory in IPv6, IPsec is one of the most suitable options for E2E security in the IoT, as mostly

IoT Layer	IoT Protocol	Security Protocol
Application	CoAP	User-defined
Transport	UDP	DTLS
Network	IPv6, RPL	IPsec, RPL security
6LoWPAN	6LoWPAN	None
Data-link	IEEE 802.15.4	802.15.4 security

Table 1.1: IoT stack with standardized security solutions.

only one application runs on a constrained device and the default security policies are enough for such scenarios. Furthermore, application developers require comparatively little effort to enable IPsec on IPv6 hosts, as it is already implemented at the network layer by device vendors.

Transport Layer: CoAP Security

Although IPsec can be used in the IoT it is not primarily designed for web protocols such as HTTP or CoAP. For web protocols Transport Layer Security (TLS) or its predecessor Secure Sockets Layer (SSL) is the most common security solution. The connection-oriented TLS protocol can only be used over stream-oriented TCP that is not the preferred method of communication for smart objects; due to lossy nature of low-power wireless networks it is hard to maintain a continuous connection in 6LoWPAN networks. An adaptation of TLS for UDP called Datagram TLS (DTLS) [19] is available. DTLS guarantees E2E security of different applications on one machine by operating between the transport and application layers. DTLS in addition to TLS that provides authentication, confidentiality, integrity, and replay protection, also provides protection against Denial of Service (DoS) attacks with the use of cookies. Though DTLS provides application level E2E security, it can only be used over the UDP protocol; TLS is used over TCP. The secure web protocol for the IoT, Secure CoAP (CoAPs), mandates the use of DTLS as the underlaying security solution for CoAP. Therefore, it is necessary to enable DTLS support in the IoT.

1.2.2 Network Security

Even with the communication security that protects the messages with confidentiality and integrity services, a number of attacks are possible against networks mainly to breach availability security services. These attacks are aimed to disrupt networks by interrupting, for example, the routing topology or by launching DoS attacks. Intrusion Detection Systems (IDS) are required to detect impostors and malicious activities in the network, and firewalls are necessary to block unauthorized access to networks. In the IoT, 6LoWPAN networks are vulnerable to a number of attacks from the Internet and from inside the network. Also, 6LoWPAN networks can become source of attacks against Internet hosts, as it is relatively easier to compromise a resource-constrained wireless node than a typical Internet host.

RPL [9], a routing protocol for low-power and lossy networks such as 6LoWPAN networks, is also prone to a number of routing attacks aimed to disrupt the topology. The IoT with 6LoWPAN networks running RPL, as shown in Figure 1.1, forms a network setup different from the typical WSNs. In the IoT, a 6BR is assumed to be always accessible, end-to-end message security is a requirement, and sensor nodes are identified by a unique IP address. In typical WSN there is no centralized manager and controller, security is usually ignored, and nodes are identifiable only within a WSN. Considering the novel characteristics of the IoT it is worth investigating the applicability of current IDS and firewall techniques in the IoT, or designing a novel IDS and firewall exploiting the contemporary IoT features and protocols.

1.2.3 Data Security

It is important to not only protect communication and networks but to also safeguard the stored sensitive data in an IoT device. Most of the IoT devices are tiny wirelessly connected resource-constrained nodes, and practically it is neither possible to physically guard each device nor to protect them with hardware-based tamper-resistant technologies such as with the use of smart cards or Trusted Platform Modules (TPM) [20]. Various software-based solutions exist that can be used to cryptographically secure stored data on nodes. For example, Codo [21] is a secure storage solution designed for the Contiki's Coffee File System [22]. There is also a need to design novel secure storage mechanisms in the context of IoT.

1.3 Research Methodology

The research methodology used in this thesis is mainly based on experimental research though analytical research is also adopted in the beginning of the thesis work. Experimental research that often starts with a concrete problem is used to evaluate the impact of one peculiar variable of a phenomenon by keeping the other variables controlled. Analytical research mainly deals with the testing of a concept that is not yet verified and specifying and inferring relationships by examining the concepts and information already available. We apply the analytical research methodology to perform a threat analysis of the WirelessHART network. We use the already known WirelessHART concepts and facts about security threats in the wireless medium and examine how the provided security mechanisms in WirelessHART guard against these threats.

Analyzing WirelessHART, a complex WSN standard, instilled me with a deep understanding of security mechanisms in low-power wireless networks and with typical limitations and issues in these networks. Based on the acquired knowledge, we develop lightweight communication, network, and data security solutions for the IoT where we mainly adapt an experimental research methodology as we have a concrete problem to solve. In order to build a communication security solution we first develop hypotheses or ideas about the architecture of IPsec, DTLS, and IEEE 802.15.4 security. We then formulate a design based on our hypothesis. To validate our hypothesis we implement and evaluate the proposed security solutions. We later examine the impact of our designed and implemented mechanisms on the IoT where we perform the evaluation of these mechanisms in a controlled experimental setup.

Realizing the need for the multi-faceted security in the IoT this thesis also provides network and data security where we develop a lightweight IDS and a novel combined secure storage and communication for the IoT. The research method we adapt here is experimental too. The first step towards solving this problem is to formulate a hypothesis, i.e., whether a novel IDS is needed for the IoT and what are the implications of a new storage model. The next step is to develop an architecture of the IDS and a secure storage mechanism that suits the IoT. To this end we provides detection techniques in the RPL-based 6LoWPAN networks and the new secure storage model. To validate our hypothesis and proposed algorithms we implement the IDS and the secure storage solution and perform extensive experiments. In the next step we analyze our experimental results that show that the proposed IDS suites the IoT and detects routing attacks in the RPL-based 6LoWPAN networks, and the new secure storage solution is more efficient than the conventional secure storage mechanisms.

1.4 Thesis Outline

This dissertation has two parts. The first part is the introduction of the thesis and second part is a collection of six papers.

Chapter 2 describes the scientific contributions of this thesis and summaries the results. Chapter 3 highlights the research contributions of this thesis and references the corresponding publications. Chapter 4 discusses the related work that motivates the need for new security solutions for the IoT. Chapter 5 concludes the thesis and provides future work; this ends the first part of the thesis.

Chapter 2

Challenges and Contributions

On one hand, constrained environments in the IoT have attributes similar to WSNs such as limited energy, processing, and storage resources, lossy wireless links, unguarded deployments, and multi-hop communication. On the other hand, the IoT is expected to have IPv6, UDP, and web support. Providing security is challenging in the Internet and in typical WSNs. It is even more challenging to enable security services in the IoT. This is because the devices are extremely heterogeneous, mostly deployed in unattended environments but closer to humans than typical WSN nodes, are globally accessible, mostly connected through lossy wireless links, require multi-hop communication, and use recent IoT protocols such as 6LoWPAN, CoAP, and RPL. This thesis provides multi-faceted security solutions for the IoT. The main contributions of this thesis are:

- It provides lightweight solutions based on standardized protocols to securely connect IoT devices. This enables the devices in the constrained environments to securely communicate with typical Internet hosts using lightweight yet standard compliant Internet security protocols such as IPsec and DTLS.
- It also contributes towards protecting 6LoWPAN networks against intrusion attempts and unauthorized access.
- In addition to communication and network security, this thesis also pro-

vides solutions to protect stored data inside a resource-constrained IoT node.

The previous chapter has highlighted security services and the standard-based security solutions in the IoT. This chapter highlights the challenges in providing security in the IoT and summarizes the contributions of this thesis.

2.1 Secure Communication: Message Security

The IoT is a hybrid network of Internet and constrained networks. Communication in the IoT can be secured with *(i)* lightweight security protocols proposed for constrained environments such as WSNs, *(ii)* novel security protocols that meet the specific requirements of the IoT, or *(iii)* established security protocols already used in the Internet. Security protocols proposed for WSNs are not designed for IP networks. Therefore, their use in the IoT requires modification of these protocols and corresponding provisioning in the current Internet. Designing novel security protocols for the IoT may result in more efficient and lightweight solutions; however, these protocols too require changes in the Internet. As the current Internet is huge, consisting of billions of devices, any security solution that requires modifications or provisions in the current Internet is not practical. It is however worth investigating the applicability of established Internet security technologies in the IoT. The primary challenge that may hinder the use of these security solutions in the IoT is that the Internet protocols are not designed for resource constrained devices but for standard computers where energy sources, processing capability, and storage space are not main constraints. One of the contributions in this thesis is to adapt the communication security protocols standardized for the Internet in the IoT, by making them lightweight yet standard compliant.

It is important that the messages in the IoT are E2E protected with confidentiality and integrity services. Also, at least integrity protection should be employed on a per-hop base in the wirelessly connected 6LoWPAN networks. Towards this end, this thesis presents the first compressed yet standard compliant IPsec for the E2E security between IoT hosts and compressed DTLS for E2E security between applications in the IoT. In order to protect messages on a per-hop base between two neighboring devices, implementation and evaluation of link layer security solutions are also provided.

***Lightweight IPsec*:** This thesis presents the first lightweight design, implementation, and evaluation of IPsec for resource-constrained devices. With 6LoW-

PAN header compression, the IPsec AH header size is reduced from 24 bytes to 16 bytes, and the ESP header size is reduced from 18 bytes to 14 bytes. This results in a lower number of bits being transmitted, more space for application data, and may avoid 6LoWPAN fragmentation; ultimately, the energy consumption is reduced as the energy consumed by radio on transmission and reception is much higher than used by microprocessor on local processing. Paper C also shows that with hardware aided crypto processing the energy overhead is further reduced by 50%. For example, when carrying 512 bytes over 4 hops, pure software-based IPsec AH involves an overhead of 26%, which is reduced to 11% with the help of hardware AES. Contrary to the common belief that IPsec is too heavy for constrained devices [3, 23], IPsec is faster than the IEEE 802.15.4 security as the number of hops grows or the data size increases. This is because the compression mechanisms substantially reduce the data overhead on fragmented traffic, and cryptographic operations are only performed at the end hosts and not at each hop as in the case of 802.15.4 security.

***Lightweight DTLS*:** Though IPsec is a feasible solution for the IoT, it is less suitable for web-based applications in the IoT. CoAP is being standardized as a web protocol for the IoT, which mandates the use of DTLS as an underlaying security solution to enable secure CoAP (CoAPs). To provide standard based E2E security in the CoAPs-enabled IoT applications, this thesis presents the first lightweight DTLS and hence CoAPs. Like IPsec, DTLS is designed for the conventional Internet and not for the resource-constrained IoT, as it is a chatty protocol and requires numerous message exchanges to establish a secure session. The DTLS header compression is based on 6LoWPAN NHC [4]. Employing these compression mechanisms significantly reduces the DTLS header sizes and ultimately results in fast and energy efficient communication compared with plain DTLS. For example, by employing the proposed mechanisms the DTLS Record header size is reduced by 62% while still maintaining the E2E standard compliance between two communication end points. The quantitative evaluation in Paper D shows that the energy overhead is significantly reduced especially when the 6LoWPAN fragmentation is employed. The use of compressed DTLS makes CoAPs considerably lightweight and a feasible security protocol for the IoT.

Realizing that smartphones with sensing capabilities, human interaction, Internet connectivity, and relatively powerful processing and storage capacities, will be an integral part of the IoT, we also provide standard-based design, implementation, and evaluation CoAPs for Android powered smartphones [24]. This paper is not included in the core contributions of this thesis.

***IEEE 802.15.4 Security*:** Prior to our work on IPsec and DTLS, 802.15.4 security was the only standard-based security solution available in 6LoWPAN networks. The IEEE 802.15.4 standard provides the link layer security to protect communication between two neighboring nodes. Link layer security is not a replacement of network or transport layer security. For 6LoWPAN networks with multiple hops, Paper C recommends that at least integrity protection should be enabled at the link layer to grand access in the wireless medium and to detect the effortless data modification attacks as early as possible. However, there is a tradeoff between the overhead of providing security at the link layer and the overhead of routing faked packets through multiple hops to the destination where they are ultimately detected. Therefore, when E2E security is provided at the network or upper layers, enabling or disabling link layer security should be carefully decided; the goal is to minimize resource usage.

In order to enable link layer security, this thesis provides an implementation of IEEE 802.15.4 security for the Contiki OS and evaluates it in a 6LoWPAN network. For 6LoWPAN networks with less hops and small data size, 802.15.4 link layer security is efficient when compared with the network layer security. Since it does not provide E2E security, the 802.15.4 security is not a replacement for IPsec or DTLS; it is therefore recommended that either IPsec or DTLS should be used in conjunction to the 802.15.4 security.

Figure 2.1 shows an IoT setup with the list of lightweight security solutions in the resource-constrained 6LoWPAN network and the corresponding plain technologies on the Internet side. The 6BR converts the compressed protocols in plain protocols and vice versa.

2.2 Secure Network: Intrusion Detection

Though communication security protects messages, networks are still vulnerable to a number of attacks aimed to disrupt the network. Intrusion Detection Systems (IDSs) and firewalls guard against such attacks. As the IoT shares characteristics with WSNs, the available IDSs for WSNs could be used in the IoT. However, most of these approaches assume that there is no centralized management and control point, no message security, and sensor nodes are uniquely identified only within WSNs. In the IoT, nodes are globally identifiable by an IP address, the 6BR is presumed to be always reachable to connect 6LoWPAN networks with the Internet, and E2E message security is a must. It is therefore worth designing a new IDS for the IoT by exploiting these novel characteristics. In spite of these characteristics, developing an IDS for the IoT

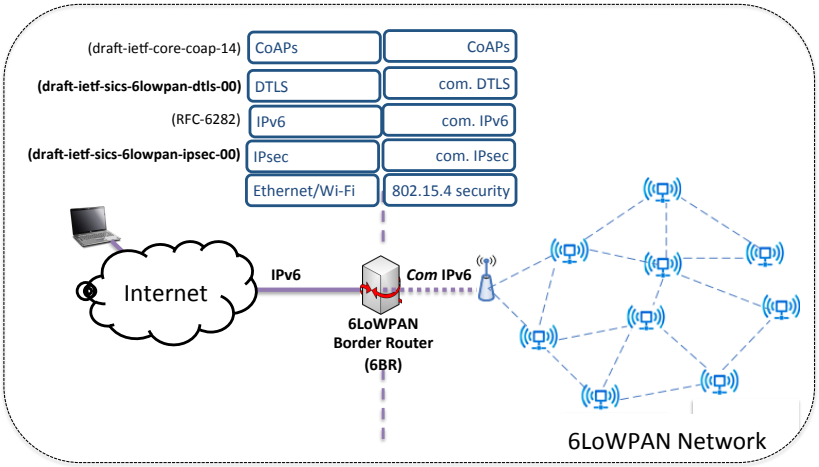


Figure 2.1: An IoT setup protected with proposed lightweight security solution, and a set of operations performed at the 6BR.

is challenging due to global accessibility, constrained resources, lossy links, and use of recent IoT protocols such as RPL.

In order to protect 6LoWPAN networks against intrusions and unwanted access this thesis provides an IDS and a mini-firewall. The IDS is designed for 6LoWPAN networks that use RPL as a routing protocol. Paper E develops a novel architecture based on a hybrid of centralized and distributed approaches. The detection algorithms in the IDS detect intrusions against RPL networks by employing contemporary lightweight detection techniques. A mini-firewall, also based on a hybrid approach, is also developed. The detection techniques are evaluated against sinkhole and selective forwarding attacks. The results show that the IDS can detect these attacks with a high true positive and detection rate. Also, the energy and ROM/RAM overhead of the IDS and the firewall are acceptable in 6LoWPAN networks.

2.3 Secure Device: Data Security

In a typical storage model, data is stored in an encrypted form along with its cryptographic hash [25], and when a remote host requests data, it is decrypted and its integrity is verified, re-encrypted and integrity protected with commu-

nication security mechanisms, and transmitted. This way the resource hungry cryptographic operations are performed twice.

With the recent advancement of flash memory, relatively more storage is now available in constrained devices. It is therefore worth exploiting the use of this additional memory in order to minimize energy consumption. Towards this end this thesis presents combined secure storage and communication mechanisms for the IoT. The proposed combined secure storage and communication mechanism, presented in Paper F, eliminates these double cryptographic operations. This work is build upon the IPv6, IPsec, and 6LoWPAN standards as a standard compliant system is more acceptable than a proprietary solution. In this new secure storage solution, data is stored on the flash file system such that it can be directly used for secure transmission. In the current design and implementation, data is protected with IPsec's ESP protocol and both the ESP header and encrypted data are stored on a flash. Prior to this operation, IP datagram header contents of future transmissions are considered in order to comply with the IPsec standard. The evaluation shows that an IP based combined secure storage and communication solution for the IoT is possible and that this can save up to 71% of a node's security related processing.

2.4 Security Analysis of WirelessHART

WirelessHART [26], though resource constrained, is a bidirectional network of relatively powerful devices and has a central network manager and controller. WirelessHART, currently the only WSN standard, designed primarily for industrial process automation and control, is well designed for other aspects than security. The provided security is spread throughout the WirelessHART specifications. The network designers and device vendors have ambiguities regarding the complete security architecture of the WirelessHART, the strength of the provided security, the security keys needed, and the functionalities and placement of Security Manager. This thesis discusses, in Paper A, the strengths and weaknesses of the provided security mechanisms in the form of a threat analysis where we analyze the WirelessHART security against the well-known threats in the wireless medium and propose recommendations to mitigate the impact of these threats. It also elaborates the functions of security manager and its placement in the network. In addition to security analysis of WirelessHART, we have also developed a WirelessHART security manager [27] and proposed secure integration of WirelessHART and legacy HART networks [28]. However, these papers are not included in the core contributions of this thesis.

The industrial community is also moving towards IP communication. This is apparent from the fact that the proposed industrial standard ISA 100.11a is IP based, and efforts are underway to apply IP communication in WirelessHART, formally named HART IP, and in ZigBee named ZigBee IP.

2.5 Standardization of Proposed Solutions

The contributions presented in this thesis mainly target HCF WirelessHART, and IETF 6LoWPAN, CoAP and RPL. During this thesis period, I attended meetings of both the HCF and IETF standardization bodies. This helped me to know the current status of the standardization efforts, to make people aware of our work, and ultimately the standardization of the work proposed in this thesis. I have attended the WirelessHART Working Group meetings in Florence and in Naples, the Internet Architecture Board (IAB) official workshop and tutorial along with the IETF 80th meeting in Prague, the IETF 83rd meeting in Paris and ETSI CoAP Plugtests. Currently, our IETF compressed IPsec draft is under review and we are working on IETF compressed DTLS draft. An ultimate aim is the inclusion of the solutions proposed in this thesis in the standard specifications. I have also published the IPsec work in the IAB workshop on Interconnecting Smart Objects with the Internet [29], and the proposed Internet Key Exchange (IKE) work in the IETF Workshop on Smart Objects Security [30].

Chapter 3

Summary of Papers

This thesis is a collection of six papers. Paper A studies the security threats in WirelessHART. Papers B-D investigate the communication security in the IoT. Paper E explores the network security in the IoT, and Paper F investigates the protection of stored data inside a node.

Paper A performs a threat analysis of WirelessHART and highlights the important security aspects of WirelessHART. Also, it stipulates the specifications of the WirelessHART security manager, its placement in the network and interactions with the other WirelessHART devices. Paper B, C, and D investigate lightweight communication security in the IoT with standard-based solutions: IPsec, DTLS, and IEEE 802.15.4. Paper E studies the protection of the IoT against network and routing attacks, and presents an IDS and firewall for RPL-based 6LoWPAN networks. Paper F explores the security of stored data inside a resource-constrained node. It presents a novel combined secure storage and communication solution for the IoT, with the special focus on minimizing cryptographic operations.

Paper A, B, and F are published in renowned international peer-reviewed conferences, Paper C and E are published in ISI indexed referenced journals, and Paper D is under submission to a journal.

3.1 Security Considerations for the WirelessHART Protocol

Shahid Raza, Adriaan Slabbert, Thiemo Voigt, Krister Landernäs. Security Considerations for the WirelessHART Protocol. *In Proceedings of 14th IEEE International Conference on Emerging Technologies and Factory Automation* (ETFA'09), September 22-26, 2009, Mallorca, Spain.

Summary

WirelessHART is a secure and reliable communication standard for industrial process automation. The WirelessHART specifications are well organized in all aspects except security: there are no separate specifications of security requirements or features. Rather, security mechanisms are described throughout the documentation. This impedes implementation of the standard and development of applications since it requires close knowledge of all the core specifications on the part of the developer.

We have thoroughly discussed the security features in the WirelessHART standard and analyzed the specified security features against the available threats in the wireless medium. We have also identified some security limitations in the standard. However, the provided security in the wireless medium, although subjected to some threats due to its wireless nature, is strong enough to be used in industrial process control environments. The physical protection of the WirelessHART devices is very important to avoid device cloning and stealing security secrets, which will lead to other security attacks. Also, the careful implementation of the Network Manager is very important. The WirelessHART standard does not enforce security in the core/wired network but the connections between the wired devices must be secured. The standard provides core security services including confidentiality, integrity, authentication, and availability; however, other security services such as non-repudiation, authorization or access control, and accounting are yet to be provided.

Contribution

In this paper we provide a comprehensive overview of WirelessHART security where we analyze the provided security mechanisms against well-known threats in the wireless medium, and propose recommendations to mitigate shortcomings. Furthermore, we elucidate the specifications of the Security Manager, its placement in the network, and interaction with the Network Manager.

My Contribution

I reviewed the WirelessHART security, performed the threat analysis of WirelessHART, and wrote the first draft of the paper.

3.2 Securing Communication in 6LoWPAN with Compressed IPsec

Shahid Raza, Simon Duquennoy, Tony Chung, Dogan Yazar, Thiemo Voigt, Utz Roedig. Securing Communication in 6LoWPAN with Compressed IPsec. *In Proceedings 7th IEEE International Conference on Distributed Computing in Sensor Systems* (DCOSS '11), June 27-29 2011, Barcelona, Spain.

Summary

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. 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.

Contribution

In this paper we provide End-to-End (E2E) secure communication between IP enabled sensor networks and the traditional Internet. We present the first compressed lightweight design, implementation, and evaluation of 6LoWPAN extension for IPsec. 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. We present the first implementation of IPsec for 6LoWPAN networks. We show that it is practical and feasible to secure WSN communication using IPsec. We evaluate the performance of our IPsec 6LoWPAN implementation in terms of code size, packet

overheads and communication performance. Our results show that the overhead comparable to the overhead of generally employed 802.15.4 link-layer security while offering the benefit of true E2E security.

My Contribution

I am the main author of the paper. I proposed the 6LoWPAN compression, contributed in implementation, and designed and performed most of the evaluation. I wrote most of the paper.

3.3 Secure Communication for the Internet of Things A Comparison of Link-Layer Security and IPsec for 6LoWPAN

Shahid Raza, Simon Duquennoy, Joel Höglund, Utz Roedig, Thiemo Voigt. Secure Communication for the Internet of Things - A Comparison of Link-Layer Security and IPsec for 6LoWPAN. *Journal of Security and Communication Networks*, DOI: 10.1002/sec.406, Early View (January 12, 2012), Wiley, 2012.

Summary

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

Contribution

In Paper B we present a 6LoWPAN/IPsec solution and perform a preliminary performance analysis of the overall system. In this paper we extend our pre-

vious work (Paper B) in several aspects. First, we describe in this paper Encapsulating Security Payload (ESP) for 6LoWPAN/IPsec while our previous work only discussed in detail the Authentication Header (AH). Second, we compare the 6LoWPAN/IPsec solution with the commonly employed 802.15.4 link-layer security, where we also implement IEEE 802.15.4 security for the Contiki OS. Third, we present a thorough testbed performance evaluation of the 6LoWPAN/IPsec solution and 802.15.4 security. We experimentally show that 6LoWPAN/IPsec outperforms 802.15.4 link-layer security as the payload size and/or the number of hops increases.

My Contribution

I designed the 6LoWPAN extension for IPsec's ESP. I implemented IEEE 802.15.4 security for the Contiki OS, and I performed most of the evaluation. I wrote the first draft of the paper.

3.4 Lithe: Lightweight Secure CoAP for the Internet of Things

Shahid Raza, Hossein Shafagh, Kasun Hewage, René Hummen, Thiemo Voigt. Lithe: Lightweight Secure CoAP for the Internet of Things. [In Submission]

Summary

CoAP enabled hosts will be an integral part of the Internet of Things (IoT). Furthermore, real world deployments of CoAP supported devices require security solutions. To this end, DTLS is the standard protocol to enable secure CoAP (CoAPs). In this paper, we investigate if the overhead of DTLS can be reduced by 6LoWPAN header compression, and present the first DTLS header compression specification for 6LoWPAN. We quantitatively show that DTLS can be compressed and its overhead can be significantly reduced using the 6LoWPAN standardized mechanisms. Our implementation and evaluation of compressed DTLS demonstrate that it is possible to reduce the CoAPs overhead, as the DTLS compression is efficient in terms of energy consumption and network- wide response time, when compared with plain CoAPs. The difference between compressed DTLS and plain DTLS is very significant, if the use of plain DTLS results in 6LoWPAN fragmentation.

Contribution

In this paper, we present Lithe- an integration of DTLS and CoAP for the IoT. With Lithe, we additionally propose a novel DTLS header compression scheme that aims to significantly reduce the header overhead of DTLS leveraging the 6LoWPAN standard. Most importantly, our proposed DTLS header compression scheme does not compromise the end-to-end security properties provided by DTLS. At the same time, it considerably reduces the number of transmitted bytes while maintaining DTLS standard compliance. The main contributions of this paper are: *(i)* we provide novel and standard compliant DTLS compression mechanisms that aim to increase the applicability of DTLS and, thus, CoAPs for constrained devices, and *(ii)* we implement the compressed DTLS in an OS for the IoT and evaluate it on real hardware; the results quantitatively show that Lithe is more efficient in many aspects than the plain CoAP/DTLS.

My Contribution

I am the main author of the paper. I proposed the compressed DTLS, and contributed in the implementation and evaluation of the compressed DTLS. I wrote most of the paper.

3.5 SVELTE: Real-time Intrusion Detection in the Internet of Things

Shahid Raza, Linus Wallgren, Thiemo Voigt. SVELTE: Real-time Intrusion Detection in the Internet of Things. *Ad Hoc Networks* Journal, Elsevier, 2013 [Accepted].

Summary

In the Internet of Things (IoT), resource-constrained things are connected to the unreliable and untrusted Internet via IPv6 and 6LoWPAN networks. Even when they are secured with encryption and authentication, these things are exposed both to wireless attacks from inside the 6LoWPAN network and from the Internet. Since these attacks may succeed, Intrusion Detection Systems (IDS) are necessary. Currently, there are no IDSs that meet the requirements of the IPv6-connected IoT since the available approaches are either customized for Wireless Sensor Networks (WSN) or for the conventional Internet. To this

end we present SVELTE, the first IDS for the IoT. We implement and evaluate SVELTE and show that it is indeed feasible to use it in the context of RPL, 6LoWPAN, and the IoT. To guard against global attacks we also design and implement a mini- firewall.

Contribution

In this paper we design, implement, and evaluate a novel intrusion detection system for the IoT that we call SVELTE. In our implementation and evaluation we primarily target routing attacks such as spoofed or altered information, sinkhole, and selective-forwarding. However, our approach can be extended to detect other attacks. We implement SVELTE in the Contiki OS and thoroughly evaluate it. Our evaluation shows that in the simulated scenarios, SVELTE detects all malicious nodes that launch our implemented sinkhole and/or selective forwarding attacks. However, the true positive rate is not 100%, i.e., we have some false alarms during the detection of malicious nodes. Also, SVELTE's overhead is small enough to deploy it on constrained nodes with limited energy and memory capacity.

My Contribution

I proposed the IDS for the IoT. I contributed in the development of the intrusion detection infrastructure, detection algorithms, and the 6Mapper. I designed the evaluation and I wrote the first draft of the paper.

3.6 Combined Secure Storage and Communication for the Internet of Things

Ibrahim Ethem Bagci, Shahid Raza, Tony Chung, Utz Roedig, Thiemo Voigt. Combined Secure Storage and Communication for the Internet of Things. *In proceedings of 10th IEEE International Conference on Sensing, Communication, and Networking* (SECON'13), June 24-27, 2013, New Orleans, USA.

Summary

The future Internet of Things (IoT) may be based on the existing and established Internet Protocol (IP). Many IoT application scenarios will handle sensitive data. However, as security requirements for storage and communication

are addressed separately, work such as key management or cryptographic processing is duplicated. Our proposed secure storage and communication framework is based on the established IPv6/6LoWPAN protocols. IPv6/6LoWPAN defines IPsec/ESP (Encapsulating Security Payload) that provides encryption and authentication of transmitted data packets. We use the same cryptographic methods and data formats defined by ESP for data processing before storage. This requires us to store not only data but also all header information that is involved in the cryptographic processing. We have shown that this is possible within the context of the IP protocol family. The described solution requires additional storage space on nodes. However, we believe that currently available flash memory sizes can absorb these overheads. Data on nodes must be secured when stored and transported in order to implement a comprehensive security solution. As resource-constrained embedded systems are limited in resources it is necessary to find efficient solutions. The proposed framework combining security aspects of storage and communication can help to achieve this goal.

Contribution

In this paper we present a framework that allows us to combine secure storage and secure communication in the IP-based IoT. We show how data can be stored securely such that it can be delivered securely upon request without further cryptographic processing. The main contributions of this paper are: *(i)* the definition of a framework for combined secure storage and communication for IP/6LoWPAN networks, *(ii)* an implementation of the framework for the Contiki operating system, and *(iii)* a detailed evaluation of the performance gains of the framework. Our prototype implementation shows that combined secure storage and communication can reduce security related real-time processing on nodes dramatically (up to 71% reduction). As shown, this can be achieved while decreasing as well a nodes power consumption (up to 32.1%).

My Contribution

I contributed in the idea of this paper, provided the initial IPsec support, and participated in writing and reviewing the paper.

Chapter 4

Related Work

There is unanimous consensus among the IoT research community that security is an important requirement in the IoT [10, 11, 31, 32, 33]. A number of security protocols has been proposed for resource-constrained WSNs [12]. However, these security protocols are often tailored to the specific application requirements and do not consider interoperability with Internet protocols. On the other hand, the IoT security protocols require interoperability with Internet protocols.

Though Garcia-Morchon et al. [31] provide general security needs in the IoT and highlight the importance of standard-based security protocols, they do not propose any adaptations in Internet protocols which make them feasible to use in the IoT. Also, no quantitative evaluation shows the applicability of standard Internet security protocols in the IoT. Granjal [32] accentuates the need for E2E security in the IoT, and shows with empirical evaluation the limitations of current sensing platforms. The community of IoT security researchers has analyzed security challenges in the IP-based IoT [33] and solutions that improve or modify standard IP security protocols that meet the requirements of resource-constrained devices. They conclude that security architectures should fit device capabilities, that proposed security protocols should ensure scalability, that cross layer interactions such as for key management is important in multi-layered solutions, and that standardization of these security solutions is important for interoperability.

4.1 Communication Security

Communication security based on End-to-End (E2E) message protection and authentication is well-recognized in the research community [10, 11, 32, 34]. Yu et al. [10] propose E2E secure communication between WSNs and Internet. They use asymmetric cryptography for key management and authentication and delegate resource hungry operations to a gateway. This leads to a need for a complex gateway, which also breaks pure E2E security between sensor nodes and hosts on Internet.

Cryptographic processing is one of the main resource hungry tasks while providing communication security. These operations include encryption and decryption, key and hash generation, and sign and verify hashes. Wander et al. [35] compare two most well-know asymmetric algorithms, RSA and Elliptic Curve Cryptography (ECC) [36], on sensor nodes and conclude that ECC is more efficient than RSA, and asymmetric cryptography is viable for constrained hardware. Later, in order to make ECC viable for WSNs, a lot of research work has focused on reducing complexity of asymmetric cryptographic algorithms, ultimately improving efficiency of key distribution protocols. For example, TinyECC [37] and NanoECC [38] use ECC in order to make cryptography feasible on resource-constrained devices. Wood et al. [39] and Hu et al. [40] have demonstrated efficient cryptography for smart objects using dedicated crypto hardware support. We have also shown that use of crypto hardware significantly reduces the overhead of cryptographic operations (Paper C). Liu et al. [41] and Chung et al. [42] describe key distribution mechanisms that save scarce bandwidth in resource constrained networks. 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. This thesis provides lightweight solutions based on standardized protocols to securely connect IoT devices.

4.1.1 IEEE 802.15.4 Security

IEEE 802.15.4 security provides standardized mechanisms for message authentication and encryption on a per-hop base in 6LoWPAN networks. 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, messages leaving the 802.15.4 network and continuing to travel on an IP network are not protected by link-layer secu-

rity mechanisms. Therefore, in many solutions, a separate security mechanism is added to protect data traveling between Internet hosts and border routers. One such example is the ArchRock PhyNET [43] that applies IPsec in tunnel mode between the border router and Internet hosts. HIP DEX [44] is another solution that can be used directly as a keying mechanism for a MAC layer security protocol. Wood et al. [39] also propose a solution to secure link-layer communication in TinyOS for IEEE 802.15.4-based WSN. Recently, Roman et al. proposed key management systems for sensor network in the context of the IoT [45] that are applicable to link-layer security. We also implement standardized 802.15.4 security for 6LoWPAN networks with hardware-aided crypto operations and show that it is viable to use 802.15.4 security in constrained environments (Paper C); however, 802.15.4 security only protects communication between two neighboring devices.

4.1.2 Transport Layer

End-to-end security can be provided by using Transport Layer Security (TLS) [46], or by its old version SSL. TLS/SSL has been proposed as a security mechanism for the IoT by Hong et al. [47]. Their evaluation shows that this security mechanism is indeed quite costly in terms of time and energy during full SSL handshake and a data packet transfer. Foulagar et al. propose a TLS implementation for smart objects [48]. However, this solution involves the border router to reduce cryptographic computational effort on smart objects and cannot be considered a full E2E solution. Brachmann et al. [49] propose TLS-DTLS mapping to protect the IoT. However, their solution requires the presence of a trusted 6BR that break E2E security at the 6BR. Kothmayr et al. [50] investigate the use of DTLS in 6LoWPANs with a Trusted Platform Module (TPM) to get hardware support for the RSA algorithm. However, in addition to specialized hardware requirement, they have used DTLS as it is without using any compression method which would shorten the lifetime of the entire network due to the redundancy in transmitted data.

Granjal et al. [34] evaluate the use of DTLS as it is with CoAP for secure communication. They note that payload space scarcity would be problematic with applications that require larger payloads. As an alternative, they suggest to employ security at other networking layers such as compressed form of IPsec. Brachmann et al. [51] provide an overview of state-of-the-art security solutions for a CoAP-based applications, and discuss the feasibility of DTSL, TLS, IPsec, or combination of these for E2E security and secure multicast communication. They assume pre-shared keys in their proposals due to

resource-constrained nature of the nodes. Recently, Koeh et al. in an IETF draft discuss the implications of securing the IP-connected IoT with DTLS [52] and propose an architecture for secure network access and management of unicast and multicast keys with extended DTLS. Garcia et al. [11] also propose and compare pre-shared based Host Identity Protocol (HIP) and DTLS as key management, secure network access, and secure communication protocols. They conclude that though HIP is efficient, it is not widely available in the current Internet; on the other hand DTLS in its current form is heavy for constrained devices and requires optimizations.

The above solutions either review the use of (D)TLS in the IoT or propose architectures that break E2E security. We reduce the overhead of DTLS in CoAP-based IoT by employing 6LoWPAN header compression mechanisms, and implement and evaluate it in an IoT setup on real hardware (Paper D). Our solution is DTLS standard complaint and ensures E2E security between CoAP applications. However, we rely on pre-shared key for initial authentication during handshake. In another work [53], we propose design ideas to reduce the overhead of the two-way certificate-based DTLS handshake. We suggest (i) pre-validation of certificates at the trusted 6BR, (ii) session resumption to avoid the overhead of a full handshake, and (iii) handshake delegation to the owner of the resource-constrained device. This work in making certificate-based authentication viable for the IoT is complementary to our work on compressed DTLS (Paper D).

Researchers are also investigating vulnerabilities in the DTLS protocol. Nadhem et al. recently demonstrated successful attacks against the DTLS protocol [54, 55].

4.1.3 IPsec

IPsec ensures the confidentiality and integrity of transport-layer headers and integrity of IP headers, which cannot be done with higher-level solutions as TLS. For these reasons, the research community [56, 57, 58] and 6LoWPAN and CoRE standardization groups [4, 59] consider IPsec a potential security solution for the IoT. On the other hand, some have regarded IPsec heavy for constrained environments [60].

We propose a standard-compliant IPsec extension for 6LoWPAN (Paper B) and evaluate it on real hardware in an IoT setup. Granjal et al. investigate the use of IPsec for 6LoWPAN [61]. However, they do not provide exact specifications of the required 6LoWPAN headers. Furthermore, no implementation is provided and no detailed evaluation of possible communication performance

is given. In their study they analyze the execution times and memory requirements of cryptographic algorithms they propose for 6LoWPAN/IPsec integration. We design, implement, and evaluate 6LoWPAN compressed IPsec for the IoT, and quantitatively compare it with the 802.15.4 security (Paper C). We propose to use IPsec in transport mode that enables E2E security between the communicating endpoints. We implement our compressed IPsec in the Contiki OS [62]. Recently, Jorge et al. [63] have extended our 6LoWPAN compressed IPsec (Paper C) and included support for IPsec in tunnel mode. They have implemented and evaluated their proposal in TinyOS.

4.1.4 Key Management in the IoT

Key Management Systems (KMSs) proposed for WSNs are tailored for specific scenario [12] and are not interoperable with Internet protocols. The KMS for the IoT should be based on standard protocols. The standard-complaint security protocol DTLS has inherited automatic KMS that the Handshake protocol provides. For key management in the resource-constrained WSNs and 6LoWPANs, pre-shared keying is still the state-of-art mechanism. Recent IETF proposal on the use of DTLS in the IoT also relies on pre-shared keys [52]. For scalable and automatic key management we have shown the viability of certificate-based DTLS in the context of IoT [53].

IPsec relies on Internet Key Exchange (IKE) [64] for key management. Kivinen proposes a lightweight IKEv2 [65] that includes the minimal set of features and does not include the optional features. This proposal too relies on shared secret for authentication and considers certificate-based authentication too heavy for the IoT. Roman et al. propose key management systems for the IoT [45] that are applicable to link-layer security. The IEEE 804.15.4 protocol does not provide a KMS. We have proposed an adaptation of the IKE that extends its key management capabilities to the IEEE 802.15.4 protocol [30]. Recently, Jennings has proposed a transitive trust provisioning for constrained devices [66], which uses a one-time password to enroll a constrained device in an IoT.

4.2 Network Security

A number of attacks against the IoT have been identified [67] in addition to those against WSN [68] that are also applicable to the IoT. Therefore, it is important to have systems that detect such attacks. The concept of intrusion detec-

tion is quite old and extensive research is carried out in this field mostly against the Internet attacks and attacks against WSN. However, no IDS are specifically designed in the context of IoT. Most of the IDS approaches for WSN are based on a distributed architecture and are built on the assumption that there is no centralized management and control point. A common IDS approach for WSNs is to utilize several special nodes distributed evenly throughout the network. These special nodes can either be physically different [69] or dynamically distributed throughout the network [70, 71]. In real deployments, however, it cannot be guaranteed that particular nodes are always present in specific locations in the network; also, the cost of employing mobile agents that move through the network might be too high. Clustering based approaches have similar issues as each cluster often requires a powerful entity for coordination [72]. The IoT has a novel architecture where the 6BR is always assumed to be accessible and is a potential place for centralized management and control.

Many IDS approaches are based upon watchdog techniques [70, 73] which could be used in the IoT. In addition to being distributed and fully deployed on sensor nodes, a general problem with watchdog-based approaches is that they require promiscuous listening, which consumes a lot of power and therefore is not suitable for constrained devices. Advanced anomaly detection approaches are proposed [74, 75], not primarily for WSNs, which on one hand can detect many intrusions efficiently but on the other hand requires intelligent learning, which is both expensive and difficult in low-power 6LoWPAN networks. Most current IDS approaches require different routing schemes that are not based on standardized mechanisms. As far as we are aware, no approach is built around 6LoWPAN and RPL in the context of the IoT. Our solution is the first design, implementation, and evaluation of the IDS for the IoT (Paper E).

4.3 Secure Storage

Solutions for secure communication and secure storage of data in the IP based IoT exist, but these functions are generally designed and operated independent of each other. There are a number of secure storage solutions available [21, 76, 77, 78]. Codo [21] is a security extension for the Coffee filesystem [22] in the Contiki OS. Codo optimizes performance of security operations by enabling caching of data for bulk encryption and decryption. We use Codo as a base and present combined secure storage and communication for the IoT, which is faster and more energy efficient than the conventional separate secure storage and communication solutions.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

The IoT is becoming a reality and serious standardization efforts are underway to interconnect the IoT devices using the IP protocols such as CoAP, 6LoWPAN, RPL, etc. IP networks will be the foundation of many services and our daily life will depend on their availability. Security is must in the IoT. Due to the sensitivity of potential applications, not just protection of communication, but a multi-faceted security is important. This thesis has presented lightweight yet standard compliant security solutions to protect communication, constrained networks, as well as stored data in devices in the IoT.

Towards secure communication in the IoT, this thesis has investigated the use of IPsec, DTLS, and the IEEE 802.15.4 security. For web-based applications in the IoT, DTLS is well suited for E2E security, and the optimized DTLS presented in this thesis has lower overhead in terms of energy consumption and response time than the plain DTLS. With currently available hardware capabilities in the IoT devices, pre-shared key authentication during the DTLS handshake is still an acceptable solution, as proposed by other recent works on DTLS as well. IPsec, mandated by IPv6, is a security solution to protect communication at the network layer, which provides security between two machines. This thesis shows that it is viable to provide E2E security in the IoT with 6LoWPAN compressed IPsec in transport mode. The evaluation

of compressed IPsec in transport mode shows that the ROM/RAM, energy, and response time overhead is acceptable. It is also shown that, contrary to the common believe, IPsec is more efficient than IEEE 802.15.4 security in 6LoWPAN networks with multiple hops and for larger message sizes. The IEEE 802.15.4 security at the link layer and upper layer security solutions (e.g. IPsec and DTLS) are not replacements for reach other. For the early identification of certain attacks (such as a data modification attack) and hence for the efficient use of network resources, in addition to E2E security solutions, link-layer security is also important in multi-hop 6LoWPAN networks.

For multi-faceted security, it is important that the IoT is protected against internal and external intrusions. Towards this end, this thesis has proposed and developed a lightweight IDS for 6LoWPAN networks that use RPL as routing protocol in the IoT. To guard against global attacks we have also designed and implemented a mini-firewall. The detection algorithms in the proposed IDS currently target spoofed or altered information, sinkhole and selective forwarding attacks. However, our IDS infrastructure is extensible and more attack detection mechanisms can be added.

Most of the IoT devices are tiny wireless devices and it is relatively easier to capture and clone them. Therefore, this thesis has also proposed a secure storage solution in the context of IoT. Unlike typical secure storage mechanisms that require separate cryptographic operations for storage and for communication security, this thesis has presented a combined secure storage and communication. Though this solution requires a little more storage space, it can reduce security related real-time processing on nodes up to 71%, and power consumption is reduced up to 32.1% when data is stored in ESP protected format.

5.2 Future Work

Pre-shared keying is still the state-of-art key management solution in the IoT. IPsec mandates pre-shared key, and CoAPs that relies on DTLS also proposes the use of pre-shared key in addition to RawPublicKey and certificate-based authentication. The communication security solutions presented in this thesis rely on shared secret key. However, with the advancement of hardware, more storage and processing capabilities with efficient energy usage are expected in the IoT devices. With these increased capabilities it may be wise to deploy certificate-based cryptography in the IoT. We have already proposed optimizations in the certificate-based authentication during the DTLS hand-

shake to make DTLS a viable solution for automatic key management, secure network access, and session negotiation. Currently, we are working on the implementation of these proposals, and plan to evaluate the full certificate-based DTLS on real hardware. Also, we plan to investigate the use of certificate-based IKEv2 for automatic key management for IPsec. In order to make IKEv2 fit for constrained environments, we have already proposed preliminary adaptations in the IKEv2. We plan to implement and evaluate the enhance IKEv2 protocol that, in addition to IPsec, also provides key management solution for the 802.15.4 security.

In the current work, we have evaluated the proposed solutions in testbeds. We plan to deploy these security technologies in real IoT deployments and evaluate them together. In parallel, we are also working on enhancements in our IDS and firewall for the IoT and extending it with more detection capabilities.

This thesis focuses on the security aspects of the IoT. Another important concern in the IoT is privacy. The importance of privacy is well-studied in the context of IoT [79, 80, 81]. However, the current work on privacy in the IoT focuses on vision, requirements, and challenges, and lacks the quantitative analysis of enabling privacy. We plan to investigate the adaptation of Privacy Enhancing Techniques (PETs) [82, 83] in the context of the IoT with empirical analysis, and plan to quantify the overhead of providing privacy in constrained environments. Some of other open security issues and challenges in the IoT are:

- Use of asymmetric cryptography with certificate-based mutual authentication in the IoT.
- Secure bootstrapping of things in the IoT with ease-of-use.
- Security and privacy of sensor data inside a cloud environment, in an integrated system of a cloud and the IoT [84, 85].
- Secure management of IoT domains.

Bibliography

[1] Adam Dunkels. Full tcp/ip for 8-bit architectures. In *Proceedings of the 1st international conference on Mobile systems, applications and services*, pages 85–98. ACM, 2003.

[2] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 1883 (Proposed Standard), December 1995. Obsoleted by RFC 2460.

[3] 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. http://www.ietf.org/rfc/rfc4919.txt.

[4] J. Hui and P. Thubert. Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks. RFC 6282, September 2011. http://www.ietf.org/rfc/rfc6282.txt.

[5] Jonathan W Hui and David E Culler. Ip is dead, long live ip for wireless sensor networks. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 15–28. ACM, 2008.

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

[7] C. Bormann. 6LoWPAN Generic Compression of Headers and Header-like Payloads, September 2012. http://tools.ietf.org/html/draft-bormann-6lowpan-ghc-05.

[8] Z. Shelby, K. Hartke, and C. Bormann. Constrained Application Protocol (CoAP), March 2013. http://tools.ietf.org/html/draft-ietf-core-coap-14.

[9] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander. RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. RFC 6550, March 2012. http://www.ietf.org/rfc/rfc6550.txt.

[10] Hong Yu, Jingsha He, Ting Zhang, Peng Xiao, and Yuqiang Zhang. Enabling end-to-end secure communication between wireless sensor networks and the internet. *World Wide Web*, pages 1–26, 2012.

[11] Oscar Garcia-Morchon, Sye Loong Keoh, Sandeep Kumar, Pedro Moreno-Sanchez, Francisco Vidal-Meca, and Jan Henrik Ziegeldorf. Securing the ip-based internet of things with hip and dtls. In *Proceedings of the sixth ACM conference on Security and privacy in wireless and mobile networks*, pages 119–124. ACM, 2013.

[12] Javier López and Jianying Zhou. *Wireless Sensor Network Security*. IOS Press, 2008.

[13] Richard E Smith. *Authentication: from passwords to public keys*. Addison-Wesley Longman Publishing Co., Inc., 2001.

[14] Naveen Sastry and David Wagner. Security considerations for ieee 802.15. 4 networks. In *Proceedings of the 3rd ACM workshop on Wireless security*, pages 32–42. ACM, 2004.

[15] S. Kent and R. Atkinson. Security architecture for the internet protocol, 1998. http://www.ietf.org/rfc/rfc2401.txt.

[16] S. Kent. IP Authentication Header. RFC 4302, 2005. http://tools.ietf.org/html/rfc4302.

[17] S. Kent. IP Encapsulating Security Payload. RFC 4303, 2005. http://tools.ietf.org/html/rfc4303.

[18] R. Atkinson. Security Architecture for the Internet Protocol. RFC 1825 (Proposed Standard), August 1995. Obsoleted by RFC 2401.

[19] E. Rescorla and N. Modadugu. Datagram Transport Layer Security Version 1.2. RFC 6347, January 2012. http://www.ietf.org/rfc/rfc6347.txt.

[20] Trusted Platform Module (TPM) Work Group. TCG specification architecture overview (TPM 2007), 2007. http://www.trustedcomputinggroup.org/.

[21] Ibrahim Ethem Bagci, Mohammad Reza Pourmirza, Shahid Raza, Utz Roedig, and Thiemo Voigt. Codo: Confidential data storage for wireless sensor networks. In *8th IEEE International Workshop on Wireless and Sensor Networks Security (WSNS 2012)*, Las Vegas, Nevada, USA, October 2012.

[22] Nicolas Tsiftes, Adam Dunkels, He Zhitao, and Thiemo Voigt. Enabling large-scale storage in sensor networks with the coffee file system. In *Proceedings of the 2009 International Conference on Information Processing in Sensor Networks*, pages 349–360, Washington, DC, USA, 2009. IEEE Computer Society.

[23] Andrey Khurri, Dmitriy Kuptsov, and Andrei Gurtov. On application of host identity protocol in wireless sensor networks. In *Mobile Adhoc and Sensor Systems (MASS), 2010 IEEE 7th International Conference on*, pages 358–345. IEEE, 2010.

[24] Thiemo Voigt Daniele Trabalza, Shahid Raza. Indigo: Secure coap for smartphones- enabling e2e secure communication in the 6iot. In *International Conference on Wireless Sensor Networks for Developing Countries (WSN4DC'13)*, pages 0–12, Jamshoro, Pakistan, April 2013.

[25] Bart Preneel. Cryptographic hash functions. *European Transactions on Telecommunications*, 5(4):431–448, 1994.

[26] Anna N. Kim, Fredrik Hekland, Stig Petersen, and Paula Doyle. When hart goes wireless: Understanding and implementing the wirelesshart standard. *IEEE International Conference on Emerging Technologies and Factory Automation*, pages 899–907, September 2008.

[27] Shahid Raza, Thiemo Voigt, Adriaan Slabbert, and Krister Landernas. Design and implementation of a security manager for wirelesshart networks. In *Proceedings of the IEEE 6th International Conference on Mobile Adhoc and Sensor Systems (IEEE MASS 2009)*, pages 995–1004, Macau, China, 2009.

[28] Shahid Raza and Thiemo Voigt. Interconnecting wirelesshart and legacy hart networks. In *Proceedings of the 6th IEEE International Conference on Distributed Computing in Sensor Systems Workshops (IEEE DCOSSW 2010)*, Santa Barbara, USA, 2010.

[29] Shahid Raza, Thiemo Voigt, and Utz Roedig. 6lowpan extension for ipsec. *Proceedings of the IETF-IAB International Workshop on Interconnecting Smart Objects with the Internet*, 2011.

[30] Shahid Raza, Thiemo Voigt, and Vilhelm Juvik. Lightweight ikev2: A key management solution for both compressed ipsec and ieee 802.15.4 security. *Proceedings of the IETF International Workshop on Smart Object Security*, March 2012.

[31] O. Garcia-Morchon, S. Keoh, S. Kumar, R. Hummen, , and R. Struik. Security Considerations in the IP-based Internet of Things, March 2013. http://tools.ietf.org/html/draft-garcia-core-security-05.

[32] Jorge Granjal, Edmundo Monteiro, and Jorge Sa Silva. On the effectiveness of end-to-end security for internet-integrated sensing applications. In *Green Computing and Communications (GreenCom), 2012 IEEE International Conference on*, pages 87–93. IEEE, 2012.

[33] Tobias Heer, Oscar Garcia-Morchon, René Hummen, Sye Loong Keoh, Sandeep S Kumar, and Klaus Wehrle. Security challenges in the ip-based internet of things. *Wireless Personal Communications*, 61(3):527–542, 2011.

[34] Jorge Granjal, Edmundo Monteiro, and Jorge Sa Silva. On the feasibility of secure application-layer communications on the web of things. In *Local Computer Networks (LCN), 2012 IEEE 37th Conference on*, pages 228–231. IEEE, 2012.

[35] Arvinderpal S Wander, Nils Gura, Hans Eberle, Vipul Gupta, and Sheueling Chang Shantz. Energy analysis of public-key cryptography for wireless sensor networks. In *Pervasive Computing and Communications, 2005. PerCom 2005. Third IEEE International Conference on*, pages 324–328. IEEE, 2005.

[36] Victor S Miller. Use of elliptic curves in cryptography. In *Advances in CryptologyCRYPTO85 Proceedings*, pages 417–426. Springer, 1986.

[37] A. Liu and P. Ning. TinyECC: A configurable library for elliptic curve cryptography in wireless sensor networks. In *7th International Conference on Information Processing in Sensor Networks (IPSN'08)*, Washington, DC, USA, 2008.

[38] P. Szczechowiak, L. Oliveira, M. Scott, M. Collier, and R. Dahab. Nanoecc: Testing the limits of elliptic curve cryptography in sensor networks. In *5th European conference on Wireless Sensor Networks (EWSN'08)*, Bologna, Italy, 2008.

[39] A. Wood and J. Stankovic. Poster abstract: AMSecure - secure link-layer communication in TinyOS for IEEE 802.15.4-based wireless sensor networks. In *4th ACM Conference on Networked Embedded Sensor Systems (SenSys'06)*, Boulder, USA, 2006.

[40] W. Hu, P. Corke, W. Shih, and L. Overs. secfleck: A public key technology platform for wireless sensor networks. In *6th European conference on Wireless Sensor Networks (EWSN'09)*, Cork, Ireland, 2009.

[41] 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.

[42] A. Chung and U. Roedig. DHB-KEY: An Efficient Key Distribution Scheme for Wireless Sensor Networks. In *4th IEEE International Workshop on Wireless and Sensor Networks Security (WSNS'08)*, Atlanta, USA, 2008.

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

[44] R. Moskowitz. HIP Diet EXchange (DEX), November 2012. http://tools.ietf.org/html/draft-moskowitz-hip-rg-dex-06.

[45] Rodrigo Roman, Cristina Alcaraz, Javier Lopez, and Nicolas Sklavos. Key management systems for sensor networks in the context of the internet of things. *Computers & Electrical Engineering*, 37(2):147–159, 2011.

[46] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard), August 2008. http://www.ietf.org/rfc/rfc5246.txt.

[47] S. Hong, D. Kim, M. Ha, S. Bae, S. J. Park, W. Jung, and J. Kim. Snail: an ip-based wireless sensor network approach to the internet of things. *Wireless Communications, IEEE*, 17(6):34–42, 2010.

[48] S. Fouladgar, B. Mainaud, K. Masmoudi, and H. Afifi. Tiny 3-tls: A trust delegation protocol for wireless sensor networks. In *3rd European Workshop on Security and Privacy in Ad-Hoc and Sensor Networks (ESAS'03)*, Hamburg, Germany, 2006.

[49] M. Brachmann, S. L. Keoh, O. G. Morchon, and S. S. Kumar. End-to-end transport security in the IP-Based Internet of Things. In *Computer Communications and Networks (ICCCN), 2012 21st International Conference on*, pages 1 –5, August 2012.

[50] T. Kothmayr, C. Schmitt, W. Hu, M. Brunig, and G. Carle. A DTLS based end-to-end security architecture for the Internet of Things with two-way authentication. In *Local Computer Networks Workshops, 2012 IEEE 37th Conference on*, pages 956–963. IEEE, 2012.

[51] Martina Brachmann, Oscar Garcia-Morchon, and Michael Kirsche. Security for practical coap applications: Issues and solution approaches. *GI/ITG KuVS Fachgesprch Sensornetze (FGSN). Universitt Stuttgart*, 2011.

[52] S. Keoh, S. Kumar, and O. Garcia-Morchon. Securing the IP-based Internet of Things with DTLS, February 2013. http://tools.ietf.org/html/draft-keoh-lwig-dtls-iot-01.

[53] R. Hummen, J. H. Ziegeldorf, H. Shafagh, S. Raza, and K. Wehrle. Making Certificate-based Authentication Viable for the Web of Things. In *Proceedings of the 2nd ACM Workshop on Hot Topics on Wireless Network Security and Privacy (HotWiSec)*, April 2013.

[54] N.J. AlFardan and K.G. Paterson. Lucky thirteen: Breaking the TLS and DTLS record protocols. In *34th IEEE Symposium on Security and Privacy*, San Francisco, California, 2013.

[55] Nadhem J AlFardan and Kenneth G Paterson. Plaintext-recovery attacks against datagram tls. In *Network and Distributed System Security Symposium (NDSS 2012)*, 2012.

[56] 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.

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

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

[59] C. Bormann. Using CoAP with IPsec, December 2012. http://tools.ietf.org/html/draft-bormann-core-ipsec-for-coap-00.

[60] C. Alcaraz, P. Najera, J. Lopez, and R. Roman. Wireless sensor networks and the internet of things: Do we need a complete integration? In *1st International Workshop on the Security of the Internet of Things (SecIoT'10)*, Tokyo, Japan, 2010.

[61] Jorge Granjal, Edmundo Monteiro, and Jorge Sá Silva. Enabling network-layer security on ipv6 wireless sensor networks. In *IEEE Global Communications Conference (GLOBECOM,10)*, Miami, USA, 2010.

[62] Adam Dunkels, Bjorn Gronvall, and Thiemo Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks*, pages 455–462. IEEE Computer Society, 2004.

[63] J. Granjal, E. Monteiro, and J. S. Silva. Network-layer security for the Internet of Things using TinyOS and BLIP. *International Journal of Communication Systems*, 2012.

[64] C. Kaufman, P. Hoffman, Y. Nir, and P. Eronen. Internet Key Exchange Protocol Version 2 (IKEv2). RFC 5996 (Proposed Standard), September 2010. http://www.ietf.org/rfc/rfc5996.txt.

[65] T. Kivinen. Minimal IKEv2, April 2013. http://tools.ietf.org/html/draft-kivinen-ipsecme-ikev2-minimal-01.

[66] C. Jennings. Transitive Trust Enrollment for Constrained Devices, April 2013. http://tools.ietf.org/html/draft-jennings-core-transitive-trust-enrollment-01.

[67] O. Garcia-Morchon, R. Hummen, S.S. Kumar, R. Struik, and S.L. Keoh. Security Considerations in the IP-based Internet of Things, March 2012. http://tools.ietf.org/html/draft-garcia-core-security-04.

[68] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. *Ad hoc networks*, 1(2):293–315, 2003.

[69] I.M. Atakli, H. Hu, Y. Chen, W.S. Ku, and Z. Su. Malicious node detection in wireless sensor networks using weighted trust evaluation. In *Proceedings of the 2008 Spring simulation multiconference*, pages 836–843. Society for Computer Simulation International, 2008.

[70] R. Roman, J. Zhou, and J. Lopez. Applying intrusion detection systems to wireless sensor networks. In *Proceedings of IEEE Consumer Communications and Networking Conference*, pages 640–644, 2006.

[71] T.H. Hai, E.N. Huh, and M. Jo. A lightweight intrusion detection framework for wireless sensor networks. *Wireless Communications and mobile computing*, 10(4):559–572, 2009.

[72] C. Rong, S. Eggen, and H. Cheng. An efficient intrusion detection scheme for wireless sensor networks. *Secure and Trust Computing, Data Management, and Applications*, 187:116–129, 2011.

[73] Sergio Marti, T. J. Giuli, Kevin Lai, and Mary Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, MobiCom '00, pages 255–265, New York, NY, USA, 2000. ACM.

[74] Amitabh Mishra, Ketan Nadkarni, and Animesh Patcha. Intrusion detection in wireless ad hoc networks. *Wireless Communications, IEEE*, 11(1):48–60, 2004.

[75] Kai Hwang, Min Cai, Ying Chen, and Min Qin. Hybrid intrusion detection with weighted signature generation over anomalous internet episodes. *Dependable and Secure Computing, IEEE Transactions on*, 4(1):41–55, 2007.

[76] Neerja Bhatnagar and Ethan L. Miller. Designing a secure reliable file system for sensor networks. In *Proceedings of the 2007 ACM workshop on Storage security and survivability*, pages 19–24, 2007.

[77] Joao Girao, Dirk Westhoff, Einar Mykletun, and Toshinori Araki. Tinypeds: Tiny persistent encrypted data storage in asynchronous wireless sensor networks. *Ad Hoc Netw.*, 5:1073–1089, September 2007.

[78] Wei Ren, Yi Ren, and Hui Zhang. Hybrids: A scheme for secure distributed data storage in wsns. In *Proceedings of the 2008 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing - Volume 02*, pages 318–323. IEEE Computer Society, 2008.

[79] Rolf H Weber. Internet of things–new security and privacy challenges. *Computer Law & Security Review*, 26(1):23–30, 2010.

[80] Vladimir Oleshchuk. Internet of things and privacy preserving technologies. In *Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology, 2009. Wireless VITAE 2009. 1st International Conference on*, pages 336–340. IEEE, 2009.

[81] Carlo Maria Medaglia and Alexandru Serbanati. An overview of privacy and security issues in the internet of things. In *The Internet of Things*, pages 389–395. Springer, 2010.

[82] Peter Langendörfer, Michael Maaser, Krzysztof Piotrowski, and Steffen Peter. Privacy enhancing techniques: A survey and classification. *Handbook of Research on Wireless Security*, 1, 2008.

[83] G De Moor, B Claerhout, and F De Meyer. Privacy enhancing techniques. *Meth Info Med*, 42:148–153, 2003.

[84] Simon Duquennoy et. al. SicsthSense. http://www.sense.sics.se.

[85] Cosm - connect to your world. https://cosm.com.

II

Included Papers

Chapter 6

Paper A: Security Considerations for the WirelessHART Protocol

Shahid Raza, Adriaan Slabbert, Thiemo Voigt, Krister Landernäs.
14th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'09), September 22-26, 2009, Mallorca, Spain.

Abstract

WirelessHART is a secure and reliable communication standard for industrial process automation. The WirelessHART specifications are well organized in all aspects except security: there are no separate specifications of security requirements or features. Rather, security mechanisms are described throughout the documentation. This impedes implementation of the standard and development of applications since it requires profound knowledge of all the core specifications on the part of the developer.

In this paper we provide a comprehensive overview of WirelessHART security: we analyze the provided security mechanisms against well known threats in the wireless medium, and propose recommendations to mitigate shortcomings. Furthermore, we elucidate the specifications of the Security Manager, its placement in the network, and interaction with the Network Manager.

6.1 Introduction

WirelessHART [1] is the first IEC approved [2] open standard for Wireless Sensor Networks (WSNs) designed primarily for industrial process automation and control systems. The applications of WirelessHART include process and equipment monitoring, environment and energy monitoring, asset management, and advanced diagnostics. The WirelessHART network is a collection of wired entities: Network Manager, Gateway, Security Manager, and Plant Automation Hosts (PAH); and wireless devices: Field devices, Adapters, Routers, Access Points, and Handheld devices [3]. The Network Manager provides overall management, network initialization functions, network scheduling and monitoring, and resource management. The Network Manager collaborates with the Security Manager for the management and distribution of security keys. The wireless devices are connected using a mesh network where each device acts as a router and must be directly connected with at least two neighboring devices to provide path diversity. The protocol stack is based on a seven layer OSI stack with additional Security and MAC sub layers. WirelessHART is a self healing and self organizing wireless protocol, in that the devices are able to find neighbors and establish paths with them, and detect network outrages and reroute.

The WirelessHART standard is developed by the HART Communication Foundation (HCF) [4] consisting of authorities in process automation and control. The WirelessHART specifications are very well designed and almost complete in all aspects except security. The provided security is spread throughout the WirelessHART specifications and the standard lacks a comprehensive document that explains and specifies the security. The network designers and device vendors encounter ambiguities regarding the complete security architecture of the WirelessHART, the strength of the provided security, the security keys needed, and the functionalities and placement of Security Manager.

The WirelessHART standard has been recently released and we are the first to analyze and clarify its security features. Our main contribution is to provide a thorough understanding of the security features in WirelessHART. We discuss the strengths and weaknesses of the provided security mechanisms in the form of threat analysis: we analyze the WirelessHART security against the well known threats in the wireless medium and propose recommendations to mitigate the impact of these threats. We also explain the security keys and their usage as the standard does not illustrate them clearly. Finally, we elaborate the functions of the Security Manager, its placement in the network, and its interaction with the Network Manager.

6.2 WirelessHART Security

The legacy HART protocol (HART 6 and earlier) uses only single parity check coding schemes [5] to detect communication errors. However, WirelessHART (HART 7) is a secure and reliable protocol for industrial automation. The field devices collect data about processes and securely send it, as an input, to other field devices. The routing information, security keys, and the timing information are sent to the devices in a secure way. In short, all data in the WirelessHART network travel in the form of WirelessHART commands and the confidentiality, integrity, and the authenticity of the commands are ensured. We can divide the provided security in the WirelessHART standard into three levels: End-to-End, Per-hop, and Peer-to-Peer.

6.2.1 End-to-End Security

End-to-end security is enforced to secure the communication between the source and destination from malevolent insiders.The Network Layer is used to provide end-to-end security; any data that is passed from the network layer to the data-link layer is enciphered (except for the NPDU header) and only the destination device is able to decipher it. All field devices in the WirelessHART network have unicast and broadcast sessions with the Gateway and Network Manager. Two field devices always communicate via the Gateway [1]. The Network Protocol Data Unit (NPDU) is shown in the Table 6.1.

NPDU Header	Security Sublayer	NPDU Payload

Table 6.1: WirelessHART Network Layer PDU

The NPDU payload in Table 6.1 is a Transport Layer PDU (TPDU) that is always encrypted using the Advanced Encryption Standard (AES) with a 128 bit key. The Security Sub-layer consists of the Message Integrity Code (MIC), the Counter, and the Security Control Byte. The NPDU header is needed for routing of data; its details can be found in the specifications [6]. The three fields in the Security Sub-layer are used as follows:

i. Security Control Byte: It is used to define the type of the security employed. The first four bits are reserved for future security enhancement

[1] It is possible to create peer-to-peer session between the two field devices but the WirelessHART standard prohibits such direct connections due to security reasons.

and the next four bits define the security types. In HART 7.1, only three types are identified, see Figure 6.1 for details.

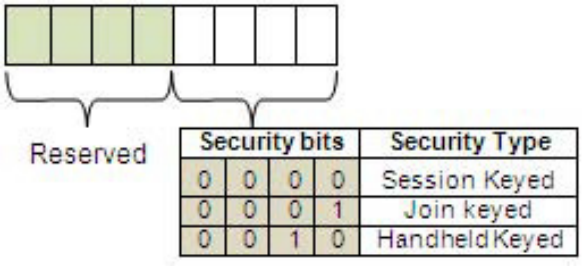


Figure 6.1: Security Control Byte

ii. Counter: A four-byte counter that is used to create the nonce.

iii. MIC: Keyed MIC is used for data integrity and source integrity (authentication) between source and destination. The MIC is calculated on the whole NPDU by setting the Time To Live (TTL), Counter, and MIC to zero. Four byte-strings are needed to calculate the MIC, including: NPDU header (*a*) - from control byte to MIC, NPDU payload (*m*) - the encrypted TPDU, *Nonce* - 13 byte long and provides defense against reply attacks, *AES key* - a 128 bit key needed for calculating the MIC. The same key is used for encrypting NPDU payload.

The Network Layer in the WirelessHART protocol stack provides three security services: confidentiality, integrity, and authentication. The AES in Counter with CBC-MAC (CCM) mode [7] is used for calculating the MIC to provide authentication and data integrity, and encrypting the NPDU payload to provide confidentiality. The same key is used for both encryption and MIC calculation. The CCM mode is the combination of *Cipher Block Chaining-Message Authentication Code* (CBC-MAC) and *Counter* modes. The two methods are highlighted below:

i. AES-CCM in CBC-MAC mode

In CBC-MAC, the message is enciphered using a block cipher algorithm in CBC mode and the last cipher block called MAC/MIC is constructed. In WirelessHART, the CBC-MAC mode is used to calculate the MIC at

the network and the data-link layers. CBC-MAC can be used for both plain text and cipher text. This mode needs the exact number of blocks and padding is used to equalize the last block. Only Encryption is used for calculating and verifying the MIC. A formatting function is applied on the unencrypted NPUD header, the encrypted NDPU payload, and the Nonce to produce the blocks B0, B1, B2...Bi; for details about this formatting function and block formation please refer to [8]. Figure 6.2 shows the operations to calculate MIC using CBC-MAC mode.

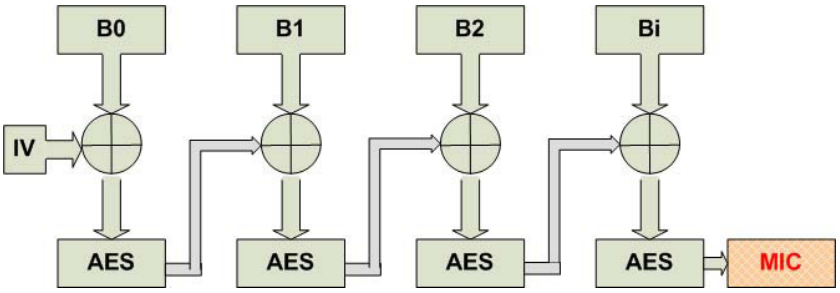


Figure 6.2: CBC-MAC mode for calculating MIC

ii. AES-CCM in Counter mode

The Counter mode is used for the encryption/decryption of the WirelessHART NPDU payload. Here too, the message blocks are created in the same fashion as above, but no padding is required and blocks can be manipulated in parallel. The cipher text C0, C1, C2,... will form an encrypted NPDU payload. The counter mode is shown in the Figure 6.3.

6.2.2 Per-Hop Security

The Data-Link Layer (DLL) is used to provide per-hop security between the two neighboring wireless devices using the Network key. Per-hop security is a defense against outsiders, i.e. devices that are not part of WirelessHART network. The Network key is known to all authenticated devices in the WirelessHART network. The keyed MIC is calculated on the entire Data Link-layer PDU (DLPDU) using the AES-CCM mode as discussed above. The four parameters for the AES CCM mode are:

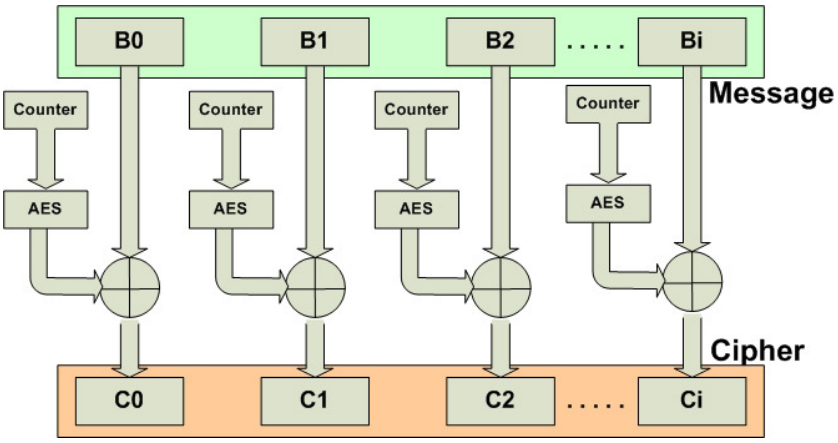


Figure 6.3: Counter mode for enciphering NPDU payload

- *m*: the encrypted message; but as the DLPDU is not encrypted the length of this byte-string is zero in WirelessHART.
- *a*: the DLPDU from 0x41 to DLPDU payload [9].
- *N*: a 13 bytes byte-string that is formed by concatenating the Absolute Slot Number (ASN) and source address [9].
- *K*: the 128 bit Network Key.

The DLL ensures source integrity (authentication) of the message between the two neighboring devices. The DLL also offers data integrity using Cyclic Redundancy Check (CRC) [2] . The WirelessHART standard uses the 16-bits ITU-T polynomial [10] to calculate the CRC.

6.2.3 Peer-to-Peer Security

All traffic in the WirelessHART network flows through the gateway, but a Handheld device can create a direct one-to-one session with the field devices using the Handheld key [3]. In order to establish such connections, the Handheld device first joins the WirelessHART network using its Join key; after

[2]CRC is not a cryptographic way to enforce integrity; rather it is a way to check communication errors.

successful joining, the Handheld device requests the Handheld key from the Network Manager. The received Handheld key is used to create a peer-to-peer session with the field device that also receives Handheld key.

Summary

The WirelessHART standard provides data confidentiality, data integrity, authentication (source integrity), and availability (using FHSS [11] and time slotting [6]) but the standard does not enforce authorization, non-repudiation, and accounting services.

6.3 Threat Analysis

A threat is an indication of a potential undesirable event [12]. The use of the wireless interface makes WirelessHART more vulnerable than legacy HART. We list possible threats against a WirelessHART network and discuss which threats are addressed by WirelessHART and which threats must be addressed. We propose recommendations to reduce the impact of the threat. The threat analysis will help developers, manufactures, and protocol designers to mitigate the impact of the threat in the design solutions.

6.3.1 Interference

Interference is an unintentional disruption of a radio signal; a signal with the same frequency and modulation technique can override the actual signal at the receiver. WirelessHART operates at the 2450 (2400-2483.5) MHz frequency band spectrum and has 16 channels; this spectrum can be shared with e.g. Wi-Fi, Bluetooth, WibRee (Bluetooth Low Energy Technology), ZigBee, and ISA100.11.a.

The WirelessHART standard uses Frequency-Hopping Spread Spectrum (FHSS) [11], uniquely assigned time slots using Time Division Multiple Access (TDMA), and path diversity which reduces the chances that interference causes actual harm to the operation of the network. With the reliability greater than 3-sigma (99.7300204%) [6] WirelessHART is the most reliable protocol among the current available solutions for industrial process automation especially if we compare it with ZigBee [13]. Nevertheless the strict and sensitive nature of a process automation system requires fail proof (100%) reliability

and failure may produce catastrophic results. The growing number of Wi-Fi, ZigBee, Bluetooth etc. devices can make the WirelessHART frequency band more vulnerable to interference in the future.

6.3.2 Jamming

Jamming is normally considered an intentional interruption of radio signal when purposely introducing noise or signal with same frequency and modulation technique as used in the target network. WirelessHART is more vulnerable to jamming attacks than interference; the attacker can deliberately introduce radio signals using commonly used Bluetooth devices like cell phones or laptops.

WirelessHART uses the concept of channel Blacklisting. If a certain frequency channel is jammed or is a continuous source of interference, then it can be blacklisted. Blacklisting enhances the reliability of the WirelessHART network but at the same time it limits the number of channels available to send/receive traffic. In spite of FHSS with 15 available channels, the active attacker can jam the WirelessHART network. The switching of channels in the FHSS is based on a pseudorandom sequence. Now if,

a. An attacker has knowledge of pseudorandom sequence (which is hard to find), he/she can calculate the actual channel. (ActualChannel = (ChannelOffset + ASN) % NumChannels) [9]

b. There are sufficient number of 2.4 GHz (Bluetooth, ZigBee, etc) devices in the range of the WirelessHART network

c. The manufacturing plant has legally deployed Wi-Fi networks in and around the WirelessHART network

d. The manufacturing plant produces sufficient amount of noise signals (which is very common there)

e. Some of the channels are already blacklisted,

then the active attacker can jam the WirelessHART network [14]. This jamming of the whole or a part of the WirelessHART network can block or even damage the machinery or plant assets.

6.3.3 Sybil

In a Sybil attack [15], an antagonist can hold multiple identities by introducing an adverse entity such as a node or piece of software into a network. The

lack of a trusted central authority in the traditional wireless ad hoc and sensor networks make it possible for the adversary to own multiple identities.

The Network Manager in the WirelessHART network binds an entity with a unique identity. The Network Manager assigns a unique Nickname to all the connected devices. Also, every device has a globally unique ID where the ID is a combination of Device Type and Device ID. The WirelessHART Gateway maintains the list of the Unique IDs and the Network Manager maintains the list of the Nicknames; the wireless devices use these Unique IDs and Nicknames along with the session keys to maintain sessions with the Gateway and Network Manager respectively. This makes Sybil attacks almost impossible in WirelessHART networks.

6.3.4 Traffic Analysis

The broadcast nature of the wireless signals make them more prone to the traffic analysis than wired signals where the attacker should be physically connected to the network.

In WirelessHART networks, the NPDU header and the whole DLPDU are unencrypted and the adversary can easily analyze the WirelessHART traffic. The NPDU header fields e.g. source/Destination addresses, Security Control byte, Nonce counter, etc. are all sent in clear. These fields provide enough information to the rival to perform analysis of the network: finding new devices by analyzing join requests, work peak hours, device usage that can help to make other attacks more effective etc.

If the DLPDU payload were allowed to be encrypted with the Network key (which is also used to calculate the MIC over DLPDU) then the traffic analysis could be minimized, but then all the intermediate devices have to decrypt the NPDU at the DLL to find the destination address, routing information, etc; this will make it difficult to meet the timing requirement of 10ms which is already hard as pointed out by Song [16]. This trade off between the security and system performance makes traffic analysis attack relatively easy.

6.3.5 DOS

Denial-of-Service (DOS) is a common attack on all networked systems; it is against the *Availability* security service. The wireless nature of WirelessHART makes it more prone to the DOS attack than legacy HART. DOS attacks against a WirelessHART network can be launched by:

- Flooding the network with join requests as the join message is encrypted with the Well-known key at the DLL.

- Sending the fake Advertisements to the neighbors (also encrypted with the Well-known key).

- Continuously modifying the DLPDU and re-computing the CRC: Now the receiving device has to verify the message integrity by calculating the MIC (as the CRC is verified); the WirelessHART protocol uses AES in CCM for calculating MIC which is an expensive operation and requires strict timing (TsTxAckDelay =1ms) requirements [16] to verify the MIC. The unverified packet will be discarded, which results in the retransmission of the packet and consumption of network resources.

- Launching a jamming attack (see section 3.2).

6.3.6 De-synchronization

The attacker can disrupt the communication between two nodes by introducing false timing information in the network and engaging the devices to waste their resources in time synchronization.

The WirelessHART standard has strict timing requirements, and the Timer [1] is one of the primary modules in the network. The Timer module has to meet the timing requirements and keep the time slots (10ms) in synchronization. The MAC sub-layer is responsible for time slotting. Each time a node receives an ACK from its time source, it adjusts its clock. The timing source for a node can be a sender [16], and if the sender is compromised it can disrupt the timing between the two nodes. Hence the participating nodes waste their resources in time synchronization.

6.3.7 Wormhole

In a wormhole attack [17] the adversary creates a tunnel between two legitimate devices by connecting them through the stronger wireless (by inaugurating radio transceivers at both ends) or wired links.

The potential WirelessHART devices that the attacker can use to launch wormhole attack are HART devices (wired) connected to WirelessHART network through the Adapters; the adversary can create a tunnel by connecting two field devices using their maintenance port. A tunnel can also be created by a wireless connection if the Network or Session keys are compromised.

WirelessHART can be subjected to wormhole attack if it uses graph routing (that supports redundant paths). However, if source routing is used then the device must use device-by-device route from source to destination. Source routing provides defense against wormhole attacks but is not reliable, since if any of the intermediate links fail a packet will be lost. One of the recommended solutions to prevent wormhole attack is packet leashing [18]. The physical protection of devices can avoid wired connected wormholes.

6.3.8 Tampering

Tampering or modification attack is the changing of stored secrets or data in transit. If the message is protected with CRC or hash, the attacker usually modifies the data and recalculates the hash or CRC. The stored secrets can be tampered by physically capturing the device and changing the data.

The WirelessHART standard uses the keyed MIC at the Network and Data-link layer to enforce integrity and provide defense against a data tampering attack. Without the knowledge of this specific key the attacker is unable to perform this attack. It is easier to perform a modification attack in the DLL than in the Network layer as the Network key is shared among all the devices and hence easy to find while session keys are device specific. Knowing the Network key and the unencrypted DLPDU, an adversary can seriously damage the normal operations of the WirelessHART network by tampering with the DLPDU and re-calculating the MIC to make it authentic.

Regular changing of the Network key is highly recommended. The physical protection of the device provides defense against the tampering of stored secrets.

6.3.9 Eavesdropping

Eavesdropping refers to the surreptitious listening of private communication. The *Confidentiality* security service is used to protect data from eavesdroppers.

The actual WirelessHART message consists of aggregated commands. These Commands, the Transport Byte, and the Device Status collectively form a NPDU payload that is encrypted with an AES 128 algorithm using unicast session key. Although some attacks [19] [20] have been identified against AES, none of them are able to crack it and AES is still a NIST USA recommended standard. For an attacker it is very hard to find a session key as it is short lived and unique for each device; hence message eavesdropping is difficult in the

WirelessHART network. Also, the use of FHSS does not allow the eavesdropper to intercept the signal without having the pseudorandom sequence [11].

6.3.10 Selective Forwarding Attack

Here the compromised node selectively drops packets; the worst form is when the node does not forward any packet and creates a black-hole [21], but normally the node selectively discards packets so that it is considered as legitimate and cannot be detected by the recovering mechanisms. The Selective Forward attack is more effective if it is backed by traffic analysis.

The Network Manager in the WirelessHART network is responsible for general monitoring of the network; the Handheld device is used to monitor the specific device. They should collectively monitor the network on regular basis to detect and eliminate these attacks. The WirelessHART command 779 (Report Device Health) can be useful in detecting this attack.

6.3.11 Exhaustion

Any device that supports the WirelessHART protocol stack and has knowledge of network parameters (Network ID, Device ID, etc.) can send messages to the neighboring devices using the Well-known key. A fake device can use the Well-known key for calculating the MIC over the DLPDU and can use a fake Join key to encrypt and authenticate the NPDU. Although this message will be discarded when received by the Network Manager (as it uses a faked Join key) it consumes network resources along the route from the field device to the Network Manager. If a series of such join attempts are initiated by an active attacker then it can give rise to a serious DOS effect/risk.

In WirelessHART networks, the attacker can only send these messages using the join slot which will not affect the communication among other networked devices. The protection of non-cryptographic secrets (Network ID, Device ID, etc.) can also eliminate this attack.

6.3.12 Spoofing

Field devices in the WirelessHART network use the Well-known key not only for joining the network but also for advertisements [3]. The adversary can spoof the new joining device by sending fake advertisements and on receipt of the

[3] WirelessHART devices have Advertisement slots that are used to publish the device presence to the new potential devices who wish to join the network.

join request it can simply discard it. If the fake device has access to the valid Network key then the spoofing attack is more effective since the device can announce its presence to the other legitimate networked devices. Moreover, this can result in a serious blockage of network traffic.

The use of different devices while joining the network can overcome this attack. The regular monitoring and changing of the Network key by the Network Manager can minimize this attack as well.

6.3.13 Collision

Collisions can occur when two or more devices try to access the same frequency channel at exactly the same time; collision can be intentional or unintentional. An attacker can also introduce collision in small portion of the packet [21].

The combination of time diversity and frequency diversity is used to minimize the collision and CRC-16 is used to detect the collision in the WirelessHART network. To *minimize* the collision, the WirelessHART protocol provides scheduled data transmission based on time slotting; TDMA and channel hopping is used to control access to the network [9]. The CRC is used to *detect* the collision based on ITU-T polynomial (aka CRC-16) [10].

The CRC-16 might not be able to detect the insertion attack (see security consideration in [10]). This attack can be avoided by better implementation and active coordination between the Physical and Data-link layer especially when the physical layer connection state changes.

6.3.14 Summary

The WirelessHART standard is secure enough to provide defense against most of the attacks. However, wormhole, de-synchronization, jamming, traffic analysis, spoofing, and exhaustion attacks need more attention.

Other than these attacks, the physical protection of the WirelessHART devices is very important. If the device is captured by the attacker it should self destruct because otherwise it can be cloned and the secret contents can be revealed. When a device is disconnected from the network, it should wipe out its volatile memory.

6.4 WirelessHART Security Manager

The Security Manager is an integral wired device in the WirelessHART network. Some of the critical points about the WirelessHART Security Manager are:

- One Security Manager can serve more than one WirelessHART network but there is only one active Security Manager per network.

- The Security manager is an application that meets the security needs of the wireless network. It can reside in a standalone device; it can be a function in the PAH; and it can be integrated in the black box consisting of Gateway, Network Manager, and Security Manager.

- The Security Manager cannot create sessions with the wireless devices; also, it is completely hidden from the Gateway.

- The interface between the Security Manager and Network Manager is not defined by the standard.

- The Security Manager provides security keys to the Network Manager that distributes them to the respective wireless devices.

Based on these prerequisites, we propose that the Security Manager should be directly connected using a dedicated link with the Network Manager at one end and with the wired/core network at the other end. This way, the Security Manager is capable of serving both the wired and wireless networks. Also, the Security Manager can serve more than one Network Manager, but the other Network Managers should be connected to the core network at one end (the other end may be connected with the Gateway). Figure 6.4 shows the placement of the Security Manager (SM) in the WirelessHART network.

According to the WirelessHART standard, the core responsibility of the Security Manager is to manage security keys. However, as a key manager the Security Manager is responsible for generation, storage, revocation, and renewal of keys. The Security Manager is not responsible for the distribution of keys to the wireless devices; instead the Security Manager provides keys to the Network Manager that in turn distributes them to the devices. The commands [22] for the key distribution are listed the Table 6.2.

As of other security functionalities, the security keys are not clearly mentioned in the WirelessHART standard and therefore we elucidate them. In a

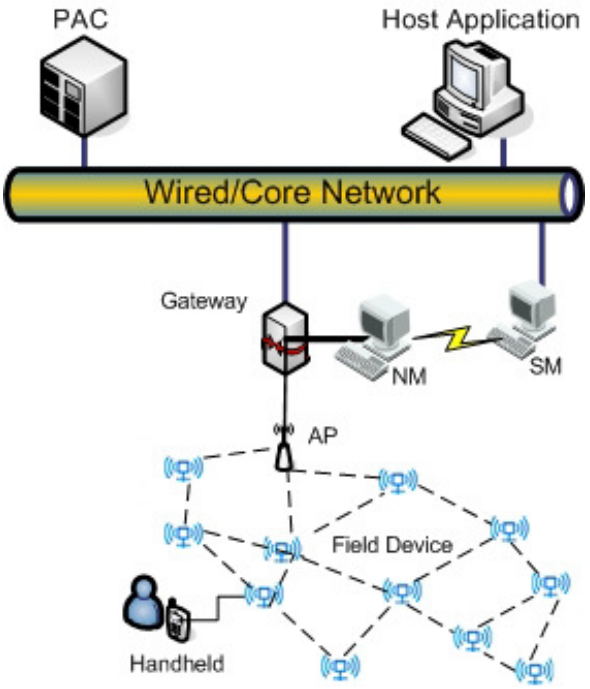


Figure 6.4: Our Proposed Placement of Security Manager in the network

WirelessHART network at maximum eight different keys can be used to encrypt/decrypt the NPDU payload and to calculate the MIC at the Network and the Data-link layer. These are:

1. *Network Key*: Used to calculate the MIC over the DLPDU. It is also used for changing the broadcast session keys.

2. *Join Key*: Used to secure the NPDU [4] during the joining process. It is also used when changing the unicast session keys both of the Network Manager and the Gateway.

3. *Unicast-NM*: Used to secure the NPDU during the communication between the Network Manager and a specific Field device. It is also used

[4] For encrypting/decrypting the NPDU payload and calculating the MIC over entire NPDU.

Keys	Commands
Session Keys	Command 963 (Write Session)
Network Key	Command 961 (Write Network Key)
Handheld Key	Command 823 (Request Session)
Join key	Command 768 (Write Join Key)

Table 6.2: Key Distribution Commands in WirelessHART

for changing the Join key.

4. *Unicast-Gateway*: Used to secure the NPDU during the communication between the Gateway and a specific Field device.

5. *Broadcast-NM*: Used to secure the NPDU during a Network Manager broadcast to all the field devices. It is also used for changing the Network key.

6. *Broadcast-Gateway*: Used to secure the NPDU during Gateway broadcast to all field devices.

7. *Handheld key*: Used to secure the NPDU during the communication between the Handheld device and the connected Field device.

8. *Well-known key*: Used to calculate the MIC over the DLPDU during the join process and while sending advertisements. The value of the Well-known key is always **772E 6861 7274 636F 6D6D 2E6F 7267.**

All wireless devices have a pre-shared Join key; the Security Manager stores all the Join keys as well. During the joining process the Network Manager asks the Security Manager for the Join key of a new joining device. This key is used to authenticate the NPDU payload and verify the MIC of the joining request. On successful authentication, all other keys are distributed to the devices.

Another important aspect the standard lacks is the interaction between the Security Manager and the Network Manager. The Security Manager manages the keys and the Network Manager uses or distributes them to the Field devices and the Gateway. The Network Manager can request a specific key from the Security Manager by providing the following parameter over a secure channel.

1. *Network ID*: As one Security Manager can serve more than one WirelessHART network each network is uniquely identified by the Network ID.

2. *Nickname*: The Network Manager maintains a list of 2-bytes Nicknames that are used to uniquely identify the WirelessHART devices. The Unique ID (UID) can be used but UIDs are maintained by the Gateway and the Security Manager cannot cannot communicate with the Gateway directly.

3. *Key Type*:The key type can be one of the seven key types listed above. The Well-known key is always the same and can be hardcoded in the Network Manager.

The WirelessHART standard does not specify the security in the wired part of the network. However, the capabilities of the Security Manager can be extended to secure the connection between the wired devices based on asymmetric or public key cryptography [23].

6.5 Security Limitations of WirelessHART

Although the WirelessHART standard is designed to be a secure and reliable protocol intended to be used for industrial process automation the current release of the standard has some security limitations. These include:

- The WirelessHART protocol does not support public key cryptography which makes it unable to provide certain security services such as non-repudiation. Strong authentication, i.e. authentication without sending the security secrets over the network is not possible either.

- No mechanisms have been specified to provide authorization and accounting security services. We need accounting when the cost of WirelessHART device is attached to its usage.

- The complete key management system is not specified; however, the commands for distribution of keys have been specified.

- Security in the wired part of the network is neither specified nor enforced.

- Secure multicast communication among the Field devices is not supported.

- Secure integration of wireless and legacy HART is not specified in the WirelessHART standard.

- The architecture of the Security Manager and the interface between the Security Manager and the Network Manager is not specified in the standard.

6.6 Conclusions and Future Work

We have thoroughly discussed the security features in the WirelessHART standard and analyzed the specified security features against the available threats in the wireless medium. We have also identified some security limitations in the standard. However, the provided security in the wireless medium, although subjected to some threats due to its wireless nature, is strong enough to be used in the industrial process control environment. The physical protection of the WirelessHART devices is very important to avoid device cloning and stealing security secrets which will lead to other security attacks. Also, the careful implementation of the Network Manager is very important. The WirelessHART standard does not enforce security in the core/wired network but the connections between the wired devices must be secured. The standard provides core security services including *Confidentiality*, *Integrity*, *Authentication*, and *Availability*; however, other necessary services such as *Non-repudiation*, *Authorization* or *Access Control*, and *Accounting* are yet to be provided.

The reserved security bits (see Security control byte [6]) can be used to enhance WirelessHART security with public key cryptography [24] [25]. Although PKI is avoided in embedded devices, having a central trusted authority (Network Manager/Security Manager) and relatively high processing power and energy resources makes WirelessHART devices different from traditional sensor devices. Research in implementing ECC and RSA on sensor nodes have shown the potential for PKI in WSNs [26]. The WirelessHART's counterpart ISA100.11.a:2008 [27] also uses public key cryptography. One way to enrich the standard with security features is to identify and specify ways to provide additional security services such as *accounting* and *access control/authorization*.

Acknowledgments

This work has been performed within the SICS Center for Networked Systems funded by VINNOVA, SSF, KKS, ABB, Ericsson, Saab Systems, TeliaSonera

and T2Data. This work has been partially supported by CONET, the Cooperating Objects Network of Excellence, funded by the European Commission under FP7 with contract number FP7-2007-2-224053.

Bibliography

[1] Anna N. Kim, Fredrik Hekland, Stig Petersen, and Paula Doyle. When hart goes wireless: Understanding and implementing the wirelesshart standard. *IEEE International Conference on Emerging Technologies and Factory Automation*, pages 899–907, September 2008.

[2] *IEC approves WirelessHART*. Control Engineering, Vol. 55 Issue 10 Pages 34-34, October 2008.

[3] *WirelessHART Device Specification,HCF_SPEC-290, Revision 1.1*. HART Communication Foundation, May 2008.

[4] *HART Communication Foundation (HCF)*. 9390 Research Blvd., Suit I-350 Austin TX 78759 USA. http://www.hartcomm2.org/index.html.

[5] Cyril Leung. Evaluation of the undetected error probability of single parity-check product codes. *IEEE Transactions on Communications*, 31(2):250–253, 1983.

[6] *Network Management Specification,HCF_SPEC-085, Revision 1.1*. HART Communication Foundation, May 2008.

[7] D. Whiting, R. Housley, and N. Ferguson. *Counter with CBC-MAC (CCM), RFC 3610*. IETF, Network Working Group, Fremont, California 94538 USA, September 2003.

[8] Morris Dworkin. *Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality*. NIST Special Publication 800-38C, May 2004.

[9] *TDMA Data Link Layer Specification, HCF_SPEC-075, Revision 1.1*. HART Communication Foundation, May 2008.

[10] W. Simpson. *PPP in HDLC Framing, RFC 1549*. IETF, Network Working Group, Fremont, California 94538 USA, December 1993.

[11] William Stallings. *Data and Computer Communications*, pages 277–282. Prentice Hall, eighth edition, 2006.

[12] Christopher Alberts and Audrey Dorofee. *Managing Information Security Risks: The OCTAVE Approach*. Addison Wesley, 09 July 2002.

[13] Tomas Lennvall, Stefan Svensson, and Fredrik Heklan. A comparison of wirelesshart and zigbee for industrial applications. *IEEE International Workshop on Factory Communication Systems*, pages 85–88, May 2008.

[14] Yee Wei Law, Marimuthu Palaniswami, Lodewijk Van Hoesel, Jeroen Doumen, Pieter Hartel, and Paul Havinga. Energy-efficient link-layer jamming attacks against wireless sensor network mac protocols. *ACM Transactions on Sensor Networks (TOSN)*, 1(5):71–80, February 2009.

[15] John R. Douceur. The sybil attack. *1st International workshop on Peer-To-Peer Systems (IPTPS)*, March 2002.

[16] Jianping Song, Song Han, Aloysius K. Mok, Deji Chen, Mike Lucas, and Mark Nixon. Wirelesshart: Applying wireless technology in real-time industrial process control. *Real-Time and Embedded Technology and Applications Symposium, 2008(RTAS-08)*, pages 377 – 386, April 2008.

[17] Levente Buttyan and Jean-Pierre Hubaux. *Security and Cooperation in Wireless Network*. Cambridge University Press, 2007.

[18] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Wormhole attacks in wireless networks. *IEEE Journal on Selected Areas in Communications*, 24(2):370– 380, February 2006.

[19] Andrey Bogdanoy. Multiple-differential side-channel collision attacks on aes, lecture notes in computer science. *10th international workshop on Cryptographic Hardware and Embedded Systems*, 5154(2):30–44, 2008.

[20] Raphael Chung-Wei Phan. Impossible differential cryptanalysis of 7-round aes. *Information Processing Letters*, 91(1):33–38, 2004.

[21] Hiran Kumar Deva Sarma and Avijit Kar. Security threats in wireless sensor networks. *IEEE A&E Systems Magazine*, June 2008.

[22] *Wireless Command Specification, HCF_SPEC-155, Revision 1.1*. HART Communication Foundation, May 2008.

[23] Harold F. Tipton and Micki Krause. *Information Security Management Handbook*, pages 1129–1135. Auerbach Publications, sixth edition, 2007.

[24] An Liu and Peng Ning. Tinyecc: A configurable library for elliptic curve cryptography in wireless sensor networks. *International Conference on Information Processing in Sensor Networks, 2008. IPSN '08*, pages 245–256, April 2008.

[25] Haodong Wang, Bo Sheng, and Qun Li. Elliptic curve cryptography-based access control in sensor networks. *International Journal of Security and Networks*, 1(3/4):127–137, 2006.

[26] Haodong Wang and Qun Li. Efficient implementation of public key cryptosystems on mote sensors. *In Proceedings of International Conference on Information and Communication Security (ICICS)*, pages 33–38, 2004.

[27] *The ISA100 Standards: Overview & Status*. http://www.isa.org/source/ISA100.11a_Release1_Status.ppt.

Chapter 7

Paper B: Securing Communication in 6LoWPAN with Compressed IPsec

Shahid Raza, Simon Duquennoy, Tony Chung, Dogan Yazar, Thiemo Voigt, Utz Roedig
7th IEEE International Conference on Distributed Computing in Sensor Systems (IEEE DCOSS '11), 27-29 June 2011, Barcelona, Spain.

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.

7.1 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.

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 7.1.

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;

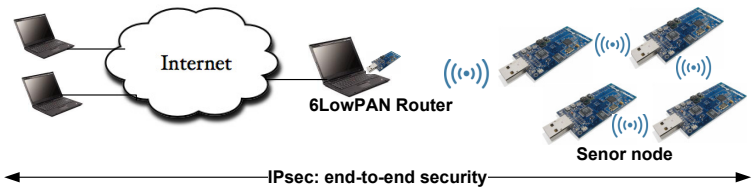


Figure 7.1: We propose to use IPsec to secure the communication between sensor nodes in 6LoWPANs and hosts in an IPv6-enabled Internet. IPsec provides E2E security using existing methods and infrastructures.

- *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 7.5 describes our proposed integration of 6LoWPAN and IPsec. After a thorough experimental evaluation of the performance of our IPsec implementation, we conclude the paper.

7.2 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 [1] and NanoEEC [2], or to simplify key distribution, for example, Liu and Ning's proposal for pairwise key predistribution [3] and DHB-KEY [4]. 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 [5] 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 [6] 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 7.3. Wood and Stankovic [7] as well as Hu et al. [8] 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 [9]. 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.

7.3 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 [10]. However, security support for IP-based low power networks is still an open issue, as mentioned in the 6LoWPAN specifications [11, 12]. 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 check-

ing. 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 [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 cannot be done with a higher-level solution like TLS. For these reasons, researchers [9, 14, 15] and 6LoWPAN standardizations groups [12] 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.

7.4 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].

7.4.1 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 [10]. This requires the use of IPv6 [16], 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 [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. 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 7.2. All hosts must support at least the hash-based message authentication code algorithm AES-XCBC-MAC-96 [20] to calculate authentication data that has a size of 12 bytes. Thus, as shown in Figure 7.2, a basic AH header has a size of 24 bytes.

ESP [19] 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

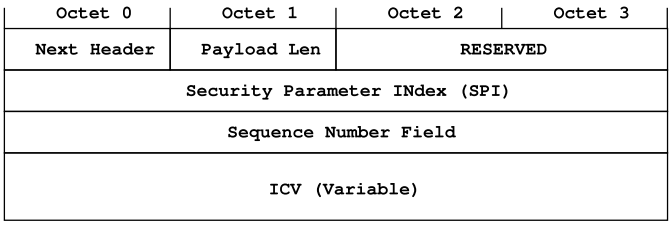


Figure 7.2: IPsec AH headers

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.

7.4.2 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 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[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 7.3.

For efficient IPv6 header compression, IPHC removes safely IPv6 header

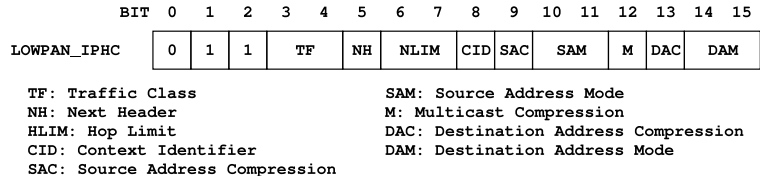


Figure 7.3: The LOWPAN_IPHC Header.

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 7.3. 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 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].

7.5 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 7.6. 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.

7.5.1 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

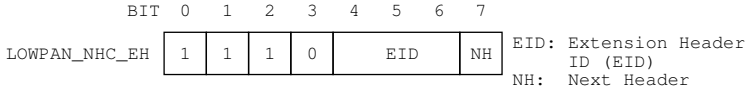


Figure 7.4: LOWPAN_NHC_EH: NHC encoding for IPv6 Extension Header

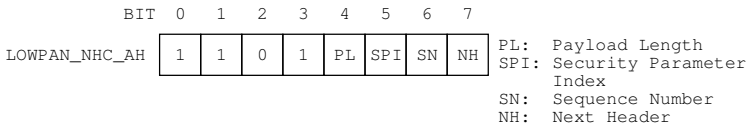


Figure 7.5: NHC_AH: NHC encoding for IPv6 Authentication Header

to encode the IPv6 Extension Header ID (EID). This NHC_EH encoding for extension headers is shown in Figure 7.4.

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.

7.5.2 LOWPAN_NHC_AH Encoding

We define the NHC encoding for the AH. Our proposed NHC for AH is shown in Figure 7.5.

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.
- 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

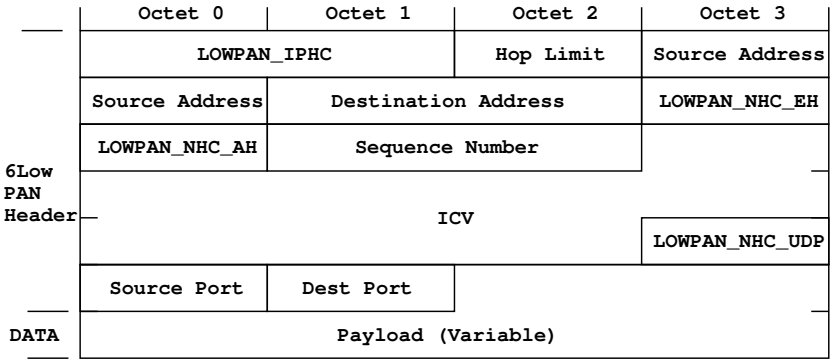


Figure 7.6: Example of a compressed IPv6/UDP packet using AH

- 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 7.6 shows compressed IPv6/UDP packet secured with AH with HMAC-SHA1-96.

7.5.3 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 7.6. Due to space limitation, we do not detail these encoding for ESP which are available in full details in a technical report [22].

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.

7.5.4 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.

7.5.5 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.

7.6 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.

7.6.1 Implementation and Experimental Setup

We implement IPsec AH and ESP for the Contiki operating system [23]. 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

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

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

NHC_EH, NHC_AH, and NHC_ESP encodings (see Section 7.5) at the SICSLoWPAN layer, the 6LoWPAN component of the μIP stack.

We use the SHA1 and AES implementations from MIRACL [24], 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 7.1 consists of four Tmote Sky [25] 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 7.1 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.

7.6.2 Memory footprint

We measure the ROM and RAM footprint of our IPsec implementation. Table 8.1 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 [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 constrained devices while leaving space for applications.

7.6.3 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 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 7.2 summarizes the packet overhead when using uncompressed IPsec, compressed IPsec and 802.15.4 link-layer security.

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.

7.6.4 Performance of Cryptography

We evaluate the efficiency of the different cryptographic algorithms and modes supported by our IPsec implementation. Figure 7.7 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

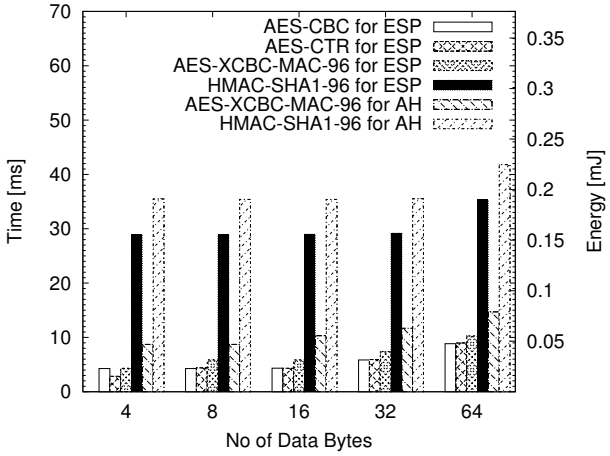


Figure 7.7: The comparison of our implemented algorithms shows that among the ones specified in the standards, AES-CBC and AES-XCBC-MAC-96 are the most efficient in terms of processing time and energy consumption. They are also mandatory and the most secure.

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 [27]. Figure 7.7 shows that these are also

7.6.5 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 7.8 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.

7.6.6 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

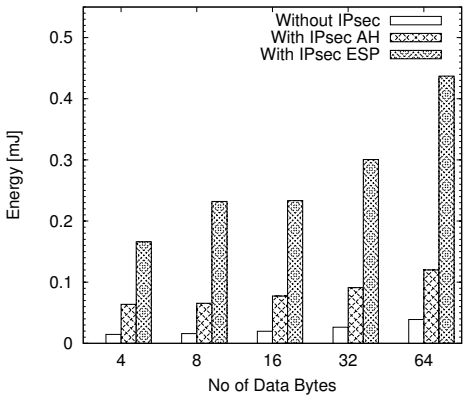
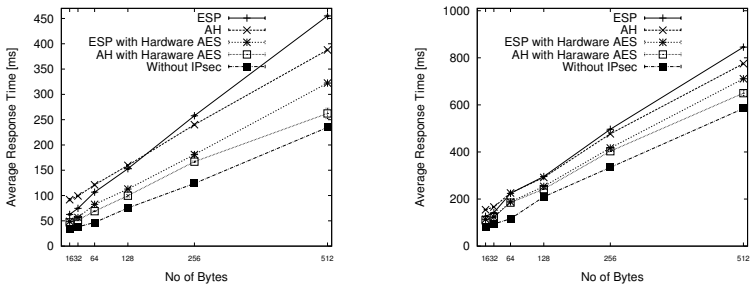


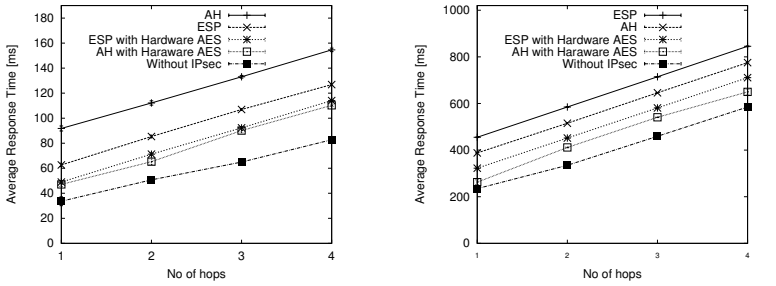
Figure 7.8: Node energy consumption is lower without IPsec and higher for ESP than for AH. Compared to other activities e.g. idle listening it is not significant.



(a) Single Hop with different data sizes

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

Figure 7.9: 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

Figure 7.10: 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.

to 4 hops and for IP datagrams with a size ranging from 16 to 512 bytes. We execute every experiment 10 times.

Figure 7.9 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 mircobenchmarks in Figure 7.7, 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 7.10 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.

7.6.7 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 inves-

tigate possible improvements we extend our prototype implementation to use this hardware for the required AES computations. Figure 7.9 and Figure 7.10 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 %.

7.7 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 preshared 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.

Bibliography

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

[2] 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.

[3] 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.

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

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

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

[7] 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.

[8] 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.

[9] 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.

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

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

[19] S. Kent. IP Encapsulating Security Payload. 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] Shahid Raza, Tony Chung, Simon Duquennoy, Dogan Yazar, Thiemo Voigt, and Utz Roedig. Securing internet of things with lightweight ipsec. Technical Report T2010:08, SICS, 2010.

[23] 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.

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

[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 *Demo session at EWSN 2007*, Delft, The Netherlands, January 2007.

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

Chapter 8

Paper C: Secure Communication for the Internet of Things - A Comparison of Link-Layer Security and IPsec for 6LoWPAN

Shahid Raza, Simon Duquennoy, Joel Höglund, Utz Roedig, Thiemo Voigt
Journal of Security and Communication Networks, DOI: 10.1002/sec.406, Early View (January 12, 201), Wiley, 2012.

Abstract

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

8.1 Introduction

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

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

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

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

Our 6LoWPAN/IPsec extension to implement true end-to-end secure communication between smart objects and Internet hosts is illustrated in Figure 8.1. We define header compression for the IPsec-related IPv6 Extension Headers: Authentication Header (AH) and Encapsulating Security Payload (ESP). We present an implementation and evaluation of our 6LoWPAN/IPsec extension for smart objects running the well known Contiki operating system [5]. This implementation exploits the cryptographic functionality provided by standard

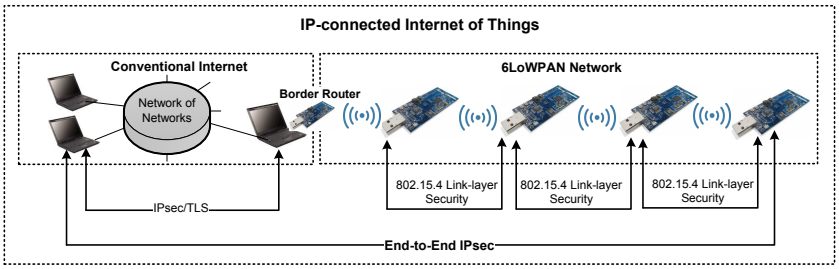


Figure 8.1: 802.15.4 security can secure communication within 6LoWPAN devices with a shared network key. IPsec/TLS can be used to secure the communication between the border router and an Internet host. We rather propose to use IPsec in an end-to-end manner, i.e., between 6LoWPAN devices and IPv6 Internet hosts.

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

The core contributions of this paper are:

- We present a definition of a 6LoWPAN extension for IPsec, supporting both Authentication Header (AH) and Encapsulating Security Payload (ESP);
- We present an implementation and a thorough testbed performance evaluation of the 6LoWPAN/IPsec extension. We show the performance gains obtained from the usage of cryptographic hardware support of 802.15.4 transceivers;
- We experimentally show that 6LoWPAN/IPsec outperforms 802.15.4

link-layer security as the payload size and/or the number of hops increases.

The next section of this paper discusses the state-of-the-art in security for the Internet of Things. Section 8.3 gives an overview of 6LoWPAN, 802.15.4 link-layer security and IPsec. Section 8.4 describes our 6LoWPAN/IPsec extension. Section 8.5 details our 6LoWPAN/IPsec and 802.15.4 link-layer security implementation for the Contiki OS. Section 8.6 presents our performance evaluation of the 6LoWPAN/IPsec extension and its comparison with the 802.15.4 security mechanisms. Section 8.7 concludes the paper.

8.2 Related Work

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

8.2.1 Embedding Cryptographic Algorithms

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

8.2.2 Securing the IoT at the Link-Layer

IP communication among smart objects uses 6LoWPAN [3] which in turn builds on the IEEE 802.15.4 [2] link-layer. 802.15.4 link-layer security is the current state-of-the-art security solution for IP-connected IoT. 802.15.4 defines data encryption and integrity verification. Benefits are network protocol independence and hardware support for the cryptographic functions by currently used 802.15.4 radio chips. Link-layer security provides *hop-by-hop* security where every node in the communication path (including the 6LoWPAN border router; see Figure 8.1) has to be trusted, and where neither host authentication

nor key management is supported. A single pre-shared key is used to protect all communication. Furthermore, messages leaving the 802.15.4 network and continuing to travel on an IP network are not protected by link-layer security mechanisms. Therefore, in many solutions, a separate security mechanism is added to protect data travelling between Internet hosts and border routers. One such example is the ArchRock PhyNET [12] that applies IPsec in tunnel mode between the border router and Internet hosts. HIP DEX [13] is another solution that can be used directly as a keying mechanism for a MAC layer security protocol. Recently, Roman et al. proposed key management systems for sensor network [14] that are applicable to link-layer security.

Since every node in the path has to be trusted, end-to-end security in the IoT cannot be achieved using the outlined approach.

8.2.3 Securing the IoT at the Transport-Layer

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

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

8.2.4 Securing the IoT at the Network-Layer

The IPsec [4] protocol suite, mandated by IPv6, provides end-to-end security for any IP communication. Like TLS and unlike link-layer solutions, it in-

cludes a key exchange mechanism and provides authentication in addition to confidentiality and integrity. By operating at the network-layer, it can be used with any transport protocol, including potential future ones. Furthermore, it ensures the confidentiality and integrity of transport-layer headers and integrity of IP headers, which cannot be done with higher-level solutions as TLS. For these reasons, the research community [18, 19, 20] and 6LoWPAN standardizations groups [3] consider IPsec a potential security solution for the IoT. On the other hand, some have regarded it as a too heavy-weight option [21].

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

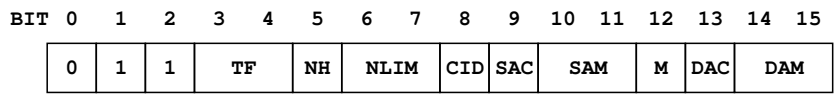
In our previous work we have presented a 6LoWPAN/IPsec solution and have made a preliminary performance analysis of the overall system [23]. In this paper we extend our previous work in several aspects. First, we describe in this paper Encapsulating Security Payload (ESP) for 6LoWPAN/IPsec while our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

8.3 Background

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

8.3.1 Overview of 6LoWPAN

IPv6 over Low-power Personal Area Network (6LoWPAN) [3] is used to tightly interconnect existing Internet and smart objects by specifying how IPv6 datagrams are to be transmitted over an IEEE 802.15.4 network. 6LoWPAN acts as a layer between the IP-layer and the link-layer that compresses IP headers and performs fragmentation when necessary. The Maximum Transmission Unit (MTU) of 802.15.4 is 127 bytes. If 802.15.4 security is enabled the maximum



TF: Traffic Class
NH: Next Header
HLIM: Hop Limit
CID: Context Identifier
SAC: Source Address Compression
SAM: Source Address Mode
M: Multicast Compression
DAC: Destination Address Compression
DAM: Destination Address Mode

(a) LOWPAN_IPHC encodings for basic IP header

(b) General Format of LOWPAN_NHC Encodings for Next Header

Figure 8.2: 6LoWPAN context-aware compression mechanisms

payload is reduced to 81 bytes, of which 40 would be consumed with uncompressed IPv6 headers. By compressing IPv6 header, 6LoWPAN increases the payload carried in 802.15.4 frames. When data cannot fit in a single frame 6LoWPAN performs fragmentation. All nodes in a 6LoWPAN network perform compression/decompression and fragmentation/reassembly. One specific node referred to as *border router* acts as a gateway between the 6LoWPAN and the conventional Internet.

The 6LoWPAN compression mechanisms [25] define header compression using LOWPAN_IPHC for IP header compression and LOWPAN_NHC for the next header compression. The IPHC [1] reduces the IP header length to 2 byte for a single hop network and 7 bytes in a multihop case. The IPHC header is shown in Figure 8.2a. The NH field when set to 1 indicates that the next header following the compressed IPv6 header is encoded with NHC. The general format of NHC is shown in Figure 8.2b. NHC has a length of 1 or more octets, where the first variable length bits identify the next header type and the remaining bits are used to encode header information. Currently, 6LoWPAN

[1] In the rest of this article LOWPAN_IPHC is referred to as IPHC and LOWPAN_NHC is referred to as NHC

defines NHC for the IP extension header and the UDP header.

8.3.2 Overview of IEEE 802.15.4 Security

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

Figure 8.3 shows the structure of a 802.15.4 packet with optional security headers. The packet overhead with link-layer security varies between 4 and 21 byte depending on the scheme used.

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

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

8.3.3 Overview of IPsec

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

IPv6 uses IPsec [4] to secure IP communication between two end points. IPsec is a collection of protocols which includes Authentication Header (AH) [27] that provides authentication services, Encapsulating Security Payload (ESP) [28] that provides both authentication and privacy services, and set of encryption and authentication algorithms [29]. A node keeps track of the so called security associations (SA) that specify how a particular IP flow should be treated in

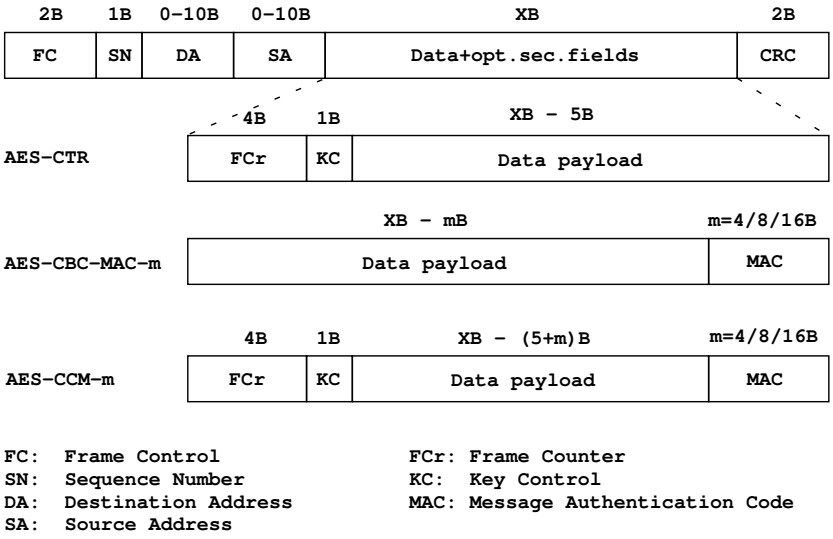


Figure 8.3: 802.15.4 frame with security headers

terms of security.

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

ESP provides origin authenticity, data integrity, and confidentiality protection of IP datagram. In contrast to AH, ESP operates on the IP payload but not on the header. ESP has common fields with AH and also contains the encrypted payload and padding required by block ciphers. ESP only encrypts payload data, padding, pad length and next header; the ICV calculation, if

selected, includes all header fields in the ESP. If we consider AES-CTR as encryption algorithm, ESP, with perfect block alignment, will have an overhead of 18 bytes (10 bytes for ESP and 8 bytes for Initialization Vector). If additional authentication using AES-XCBC-MAC-96 is used the ESP size grows to 30 bytes, as the minimum length of AES-XCBC-MAC-96 is 12 bytes.

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

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

8.4 6LoWPAN/IPsec Extension

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

8.4.1 LOWPAN_NHC Extension Header Encoding

As discussed in the background section the 6LoWPAN draft defines the general format of NHC that can be used to encode IP next header. We define NHC encodings for the two IP extension headers namely AH and ESP. 6LoWPAN already defines NHC encodings for IP extension headers (NHC_EH) that can be used to link AH and ESP extension headers. NHC encodings for the IPv6 Extension Headers consist of a NHC octet where three bits (bits 4, 5 and 6) are used to encode the IPv6 Extension Header ID (EID). The NHC_EH encoding for extension headers is shown in Figure 8.4.

Out of eight possible values for the EID, six are assigned by the HC15 [25] specification. The remaining two slots (101 and 110) are currently reserved.

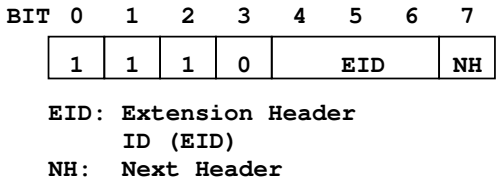


Figure 8.4: LOWPAN_NHC_EH: NHC encoding for IPv6 Extension Header

As AH and ESP are IP extension headers it makes sense to use one of these reserved slots for AH and ESP compression. We propose to use one of the reserved slots, say 101, to identify that the next header is an AH or ESP header. The ID bits in the proposed NHC for AH and ESP identify that the current header is AH or ESP, see Section 8.4.2 and 8.4.3. It is also necessary to set the last bit in NHC_EH to 1 to specify that the next header is NHC encoded.

8.4.2 LOWPAN_NHC_AH Encoding

Figure 8.6 describes our NHC encoding for AH. Next, we describe the role of all fields:

- The first four bits in the NHC_AH represent the NHC ID we define for AH. These are set to 1101.
- If $PL = 0$: The payload length (length of the IPsec header) field in AH is omitted. This length can be obtained from the SPI value because the length of the authenticating data 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 802.15.4 network is used and the SPI field is omitted. We set the default SPI value to 1. This does not mean that all nodes use the same 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.
- If $SN = 0$: First 16 bits of sequence number are elided. The remaining bits are carried inline.

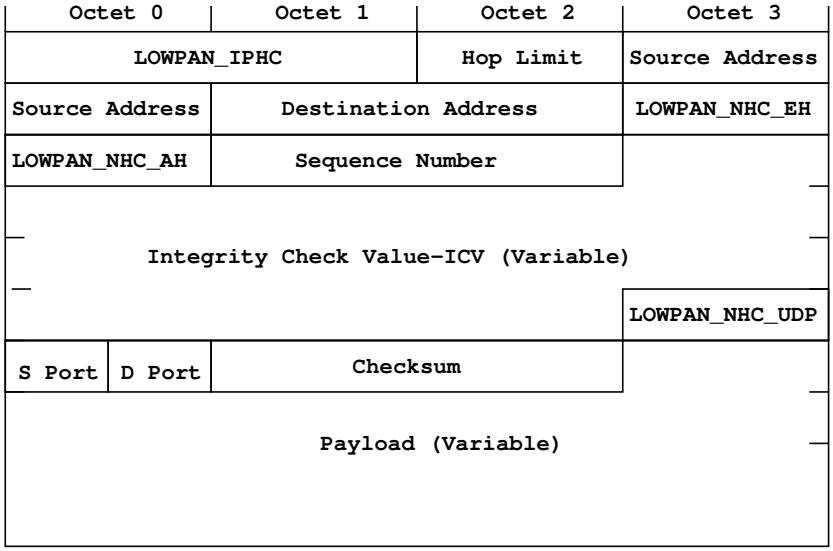


Figure 8.5: A compressed and AH secured IPv6/UDP packet.

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

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

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

Note that even when used in 6LoWPAN, AH calculates the MIC on the uncompressed IP header, thus allowing authenticated communication with Internet hosts. The minimum length of a standard AH, supporting the mandatory HMAC-SHA1-96 and AES-XCBC-MAC-96, consists of 12 bytes of header fields plus 12 bytes of ICV. After optimal compression we obtain a header size of 4 byte plus 12 bytes of ICV. Figure 8.5 shows compressed IPv6/UDP packet secured with AH using HMAC-SHA1-96.

8.4.3 LOWPAN_NHC_ESP Encoding

Figure 8.8 shows the NHC encodings we propose for ESP. Next, we describe the function of each header field:

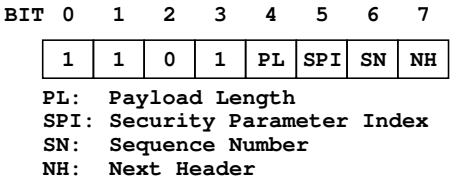


Figure 8.6: LOWPAN_NHC_AH: NHC encodings for IPv6 AH.

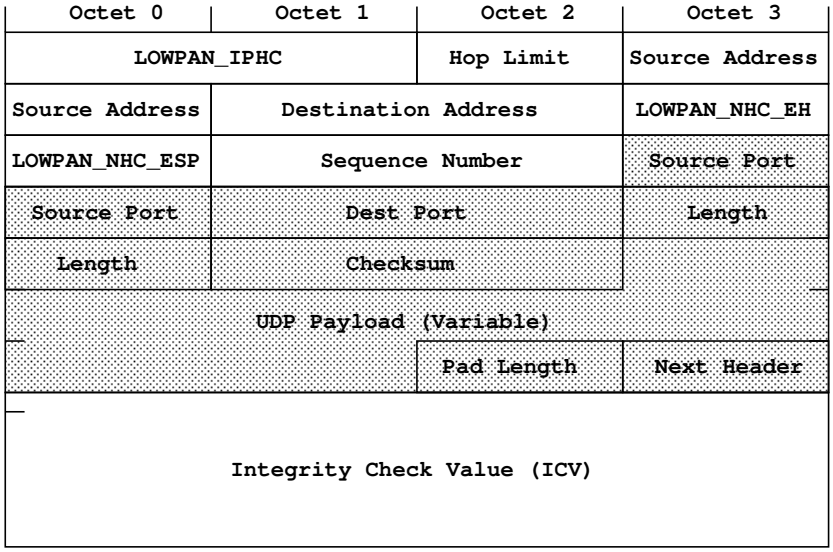


Figure 8.7: A compressed and ESP secured IPv6/UDP packet.

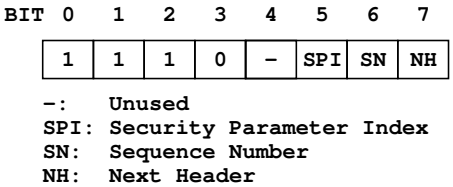


Figure 8.8: LOWPAN_NHC_ESP: NHC encoding for IPv6 ESP.

- The first four bits in the NHC_ESP represent the NHC ID we define for ESP. These are set to 1110.

- The next bit is unused. We leave this field empty to achieve coding similarity between AH and ESP (ESP does not have a payload length field). However, this field could be used to increase SPI coding to two bits if required.

- If $SPI = 0$: the default SPI for the 802.15.4 network is used and the SPI field is omitted. We set the default SPI value to 1. This does not mean that all nodes use the same 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.

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

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

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

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

Recall that the minimum ESP overhead without authentication, AES-CBC and perfect block alignment is 18 bytes. After optimal compression this header overhead is reduced to 14 bytes. ESP with authentication contains additional 12 bytes of ICV. Figure 8.7 shows an UDP/IP packet secured with compressed ESP. The shaded portion represents cipher-text.

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

8.5 Implementation

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

8.5.1 Link-layer Security Implementation

Our IEEE 802.15.4 link-layer security implementation supports header construction for all security modes described in the standard. The construction of the 802.15.4 frame is performed in software while the cryptographic operations are performed by the CC2420 radio chip. Manual key distribution is used as key management is not standardized in 802.15.4. Hence, one pre-defined key is used on all nodes in the network.

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

8.5.2 IPsec Implementation

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

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

We propose an alternate implementation of AES-based modes of operation leveraging the cryptographic capabilities of the CC2420 radio chip. Though the mode of operations are still implemented in software, they rely on the CC2420

to perform 128-bit AES block encryption/decryption. We use this hardware-aided operation for both authentication and encryption in IPsec.

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

8.5.3 Concurrent Use

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

8.6 Evaluation and Results

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

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

Table 8.1: Memory footprints show that AH and ESP consume just 1.5Kb and 4.4Kb for standard recommended IPsec algorithms using hardware-aided AES.

8.6.1 Experimental Setup

Our experimental setup shown in Figure 8.1 consists of four Tmote Sky [33] nodes, one Tmote Sky acting as a 6LoWPAN border router and a Internet host running Ubuntu OS. The Internet host runs the IPsec stack of the Linux kernel (version 2.6.38). The four smart objects (nodes) on the right side in Figure 8.1 form a multihop network. Using this setup we test different security configurations. In all experiments the nodes execute a single application that listens to a fixed UDP port. Messages are generated by a client program on the Linux host and are forwarded over the multihop network to the destination node. Depending on the experiment different nodes on the 6LoWPAN network are selected as destination. Every node needs to process packets using the 6LoWPAN-layer and μIP. If IPsec is used the Linux host and the destination node must perform cryptographic processing. If link-layer security is used all wireless nodes in the transmission path are involved in cryptographic processing. The message payload is forwarded to the application on the node that generates a new datagram of the same size as response. This reply is sent back to the Linux host following the reverse transmission path. For some of the experiments, IPsec is used to provide secure end-to-end (E2E) communication between the destination node and the Internet host, while in other experiments hop-to-hop security is provided at the link-layer.

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

8.6.2 Memory Footprint Comparison

We measure the ROM and RAM footprint of our link-layer security and IPsec implementation (see Table 8.1). The memory footprints are compared with a reference Contiki system including μIP and SICSLoWPAN.

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

The memory footprint overhead of the IPsec implementation is comparatively larger as most cryptographic algorithms have to be implemented in software. However, with hardware AES support the overhead is significantly decreased. A pure software implementation of ESP with AES-CBC + AES-XCBC-MAC consumes 35.3 Kb of ROM that can be reduced to 30.5 Kb with hardware support. Table 8.1 shows a full comparison of memory footprints with no security, with link-layer security, with software implemented IPsec, and with hybrid implementation of IPsec when AES is performed in hardware and modes of operations are performed in software. It is worth mentioning that unlike AES-CBC, the AES-CTR mode of operation only relies on AES encryption. Thus, the AES-CTR + AES-XCBC-MAC-96 configuration can be implemented without AES decryption, resulting in a particularly low memory footprint. The ROM additional overhead is 2.8 Kb that is comparable with the 1.2 Kb overhead of the link-layer footprint. The RAM footprint is calculated as the sum of the global data segments and the runtime stack usage that we measure by running Contiki in the MSPSim emulator [36]. With an additional footprint of 1.1 kB, the AH HMAC-SHA1 configuration is the most RAM-consuming configuration. When using other modes of operation, the RAM usage lies between only 0.3 and 0.5 kB.

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

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

Table 8.2: The header overhead of compressed IPsec is slightly larger than that of 802.15.4, while IPsec provides end-to-end security. As soon as fragmentation is needed, IPsec has a lower header overhead than link-layer security.

8.6.3 Header Overhead Comparison

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

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

As IPsec operates at the network-layer, its header is added to every IP datagram. However, no additional header related to security has to be added to potential 6LoWPAN link-layer fragments. When using 802.15.4 link-layer security, every frame is secured independently. Thus, security related header information has to be added to every single 6LoWPAN link-layer fragment and the overall header overhead is dependent on the number of fragments transported. Table 8.2 shows the per message header overhead incurred using IPsec and link-layer security for different security services.

Integrity Only When using link-layer security, the packet overhead for the authentication scheme is exactly the length of the MAC, i.e. 12 bytes per fragment when using AES-CBC-MAC-96. With IPsec and AES-XCBC-MAC-96, the AH header involves an overhead of 24 bytes per IP packet. Thanks to the IPsec header compression we defined, this overhead is reduced to 16 bytes. As soon as the IP datagram requires more than a single fragment, IPsec provides a lower header overhead than link-layer security.

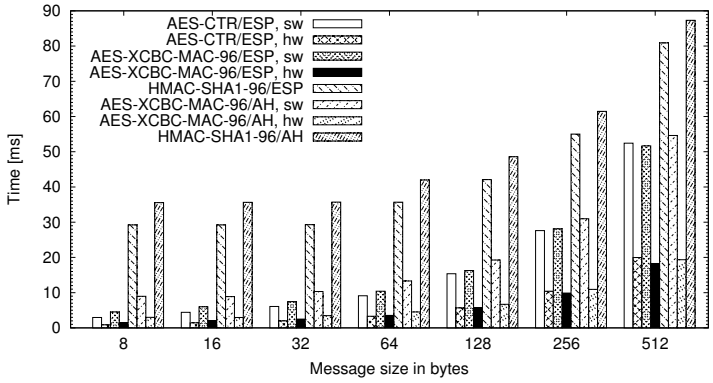


Figure 8.9: Micro-benchmarks for the supported security algorithms. Also pure link-layer encryption was evaluated but is not shown, as the maximum overhead, the time for 512 bytes, was 0.27 ms, which would not be visible in the graph. The comparison shows that AES-CTR and AES-XCBC-MAC-96 are the most efficient in terms of processing time and energy consumption, especially if hardware encryption is available. Together they provide the best level of security among the standardized algorithms.

Confidentiality Only If only message encryption is required, the 802.15.4 link-layer security provides AES-CTR which has a 5 bytes overhead per fragment. In comparison, IPsec with ESP and AES-CBC leads to an overhead of 18 bytes per packet, reduced to 14 bytes thanks to header compression. Here, the data overhead of IPsec is amortized for datagrams of 3 or more fragments.

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

8.6.4 Evaluation of Cryptographic Algorithms

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

The purely hardware based encryption used for link-layer security is extremely efficient. Compared to IPsec, processing times and energy consumption are very small when looking at a single node. The link-layer processing time for a message of 512 bytes is less than 0.3 ms which corresponds to an energy consumption of 2 μJ. Figure 8.9 shows the IPsec overhead in time for different message sizes, with or without hardware encryption. The authentication algorithms are compared separately for AH and ESP: with AH the MAC is calculated over the IP header and payload packet, while in ESP the IP header is neither encrypted nor authenticated.

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

The efficiency of IPsec calculation can be improved by employing cryptographic functions provided by node hardware. For example, the CC2420 radio chip present on many node platforms provides such functionality. This allows part of the encryption and decryption to be done faster, using less energy. In the benchmark it can be seen that by using hardware encryption, the overhead is reduced with up to 60% in time, with corresponding reduction in energy.

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

8.6.5 Energy Consumption Comparison

Securing the Internet of Things has a cost in terms of added energy usage. We measure the energy overhead using 802.15.4 link-layer security with AES-CCM and IPsec with AES-CTR + AES-XCBC-MAC. Measurements are carried out on a Tmote Sky using Powertrace, Contiki's integrated power profiler

[38]. To provide a fair comparison across different experiments, we consider a constant voltage of 3V, and we report energy rather than mote lifetime – lifetime cannot be estimated accurately because the way battery discharge varies with the environmental settings and across different instances of a same hardware [39].

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

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

Figure 8.10 shows the energy consumption of a client node using only link-layer security, IPsec using either AH or ESP, and with no security. The results show that the energy consumption with IPsec is clearly increased compared with using no security, or when just using link-layer security. Still we argue that the overhead is acceptable in cases where end-to-end security is needed. Moreover, the overhead is small if compared with the energy consumption of the radio chip when it is turned on. In a radio duty cycled network with one percent listening time, in less than 2 sec of listening the radio consumes as much energy as the maximum measured security overhead, around 1.2 mJ for ESP software processing of a 512 byte message.

Whenever supported by the hardware, the overhead can be reduced by employing hardware aided cryptographic processing. As can be seen in the Figure 8.10, hardware aided encryption reduces the energy consumption by 50%.

Although the IPsec overhead seems large for a destination node, we get a different picture if we focus on a forwarding node. With IPsec, only end-nodes have to perform security operations. With link-layer security, every node in the path has to process encryption/decryption and/or authentication, because it needs to access (protected) IP header fields in order to perform routing. As a result, IPsec has no overhead for forwarding nodes, besides a small added amount of 6LoWPAN header processing. For link-layer security on the other hand, the energy consumption grows with every node on the path between two

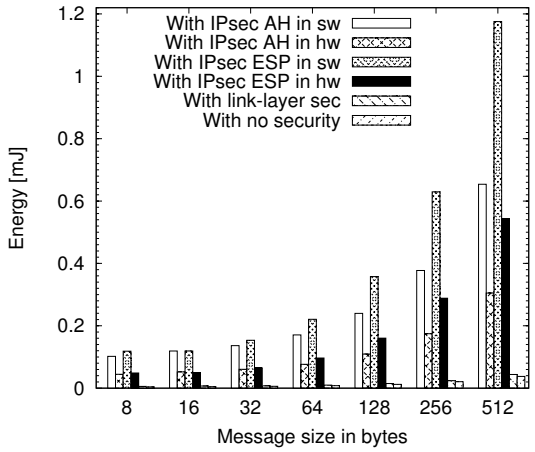


Figure 8.10: The node energy consumption is lower without IPsec and higher for ESP than for AH.

end-nodes. Unlike link-layer security, IPsec can thereby be used together with nodes that do not support any security at all. The measurements for nodes acting as forwarders in the multihop network are consistent with the micro-benchmarks, with a security processing time overhead of 0.3 ms for forwarding a 512 bytes message. This does not include the radio transmission times - added link-layer security headers can cause a message to be fragmented in more packets, in which case the system total energy overhead will increase noticeably. This is further discussed below.

8.6.6 Overall Network Performance

We evaluate the system response time for different security solutions and services with different network diameters and different payload sizes. The response time in this case is the time an IP datagram takes to travel from an IP connected Linux machine to a smart object plus the travel time of the response back to the sender. We let the size of the IP datagrams vary between 16 and 512 bytes. The number of hops ranges from 1 to 4 hops. We conduct every experiment 10 times and display the mean values using error bars for the standard deviation. We measure response time with 802.15.4 link-layer security enabled, with IPsec, and with no security. There are different modes of opera-

tion in both IPsec and link-layer security which correspond to different security services. We present our results in accordance with the security services needs.

Performance Impact of Integrity

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

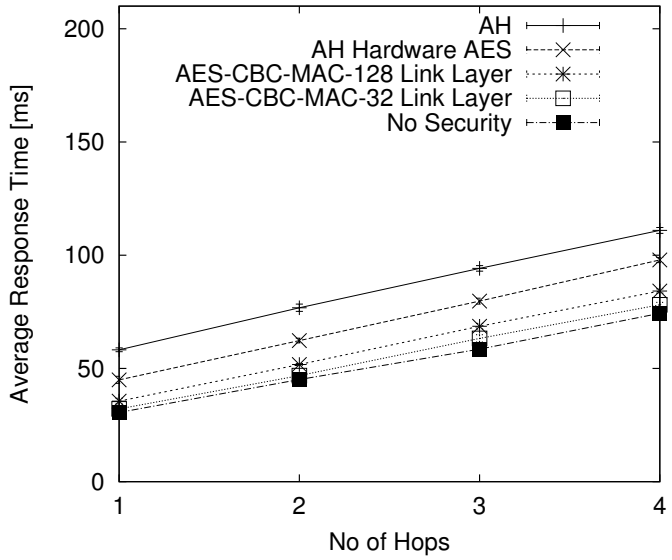
Figure 8.11 shows the response time in relation to the network diameter for link-layer AES-CBC-MAC with minimum and maximum MIC sizes, IPsec AH with XCBC-MAC-SHA1-96, and without security. For small data sizes the overhead of link-layer security is negligible. When carrying 512 bytes over 4 hops, pure software AH involves an overhead of 26 %, which is reduced to 11 % with the help of hardware AES. A point worth mentioning is that for a given data size, the overhead of IPsec is constant over the number of hops. For intermediate nodes the cost of forwarding the data with and without IPsec is the same; the overhead is only incurred for computation on the end nodes. With large data sizes (512 bytes), the overhead of link-layer security increases with the number of hops, due to the per-fragment header overhead. As a result, with 2 hops or more, AH is faster than link-layer security.

Figure 8.12 shows the response time, depending on the IP payload size, of link-layer AES-CBC-MAC with minimum and maximum MIC sizes, IPsec AH overhead with XCBC-MAC-SHA1-96, and with no security. Consistently with the mirco-benchmarks in Figure 8.9, the overhead of IPsec grows linearly with datagram sizes. In the 4-hops case, we observe that hardware-aided AH is faster than link-layer security for data sizes of 256 bytes and more.

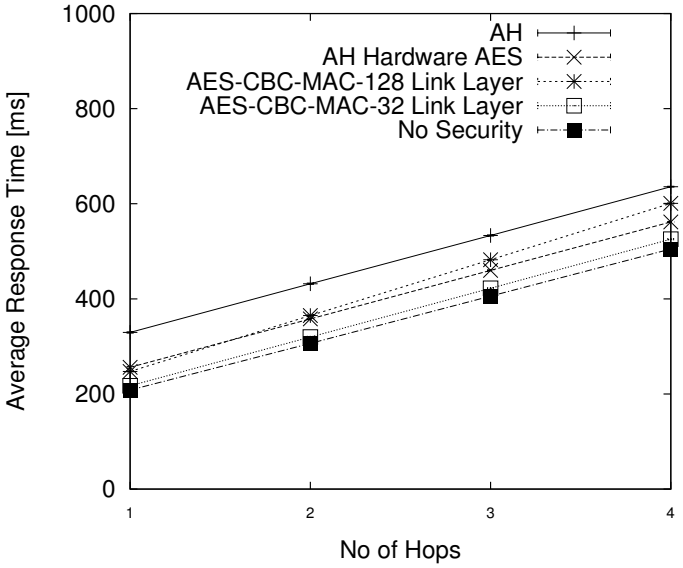
Performance Impact of Integrity and Confidentiality

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

Figures 8.13 and 8.14 show the response times obtained with different IPsec, link-layer security and with no security. For small data sizes, the overhead of link-layer security is very small. Over 4 hops and with a data size of 512 bytes, ESP incurs an overhead of 46 %, lowered to 19 % with hardware-aided AES. As the number of hops grows or the data size increases, ESP be-

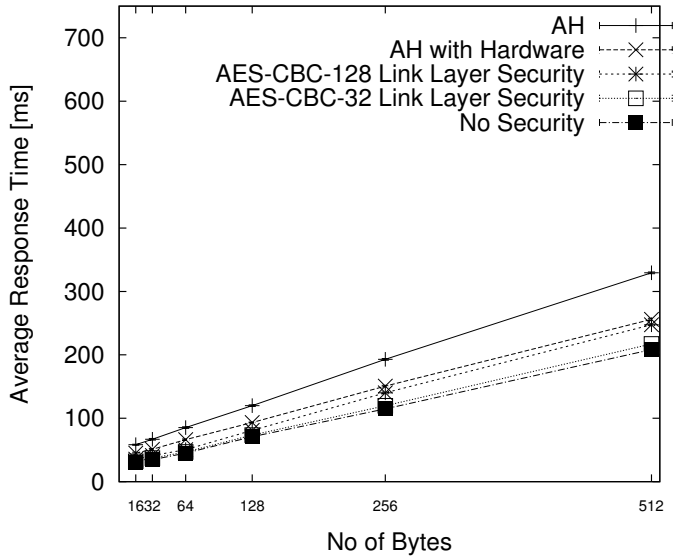


(a) Multi hops with 16 bytes data size

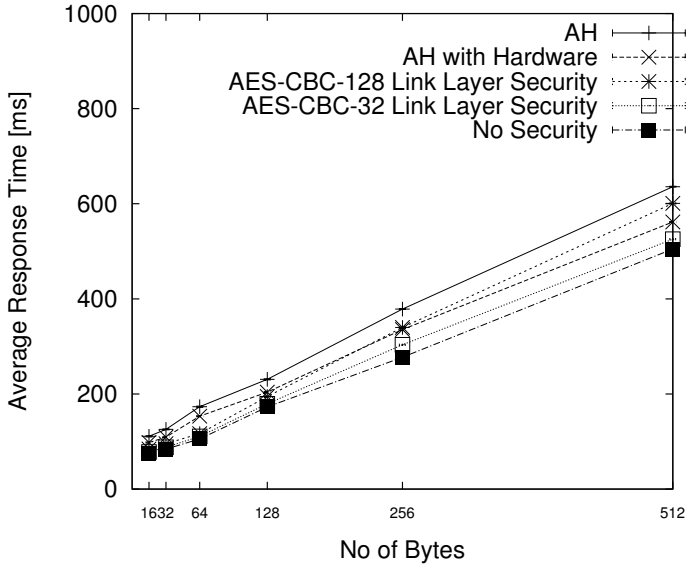


(b) Multi hops with 512 byte data size

Figure 8.11: Response time versus number of hops when *integrity* is enabled. The 802.15.4 link-layer security is better for small data sizes while IPsec gets better for large data sizes across multiple hops. The overhead of IPsec is constant across a single hop and a multihop network while link-layer security overhead increases with the number of hops.

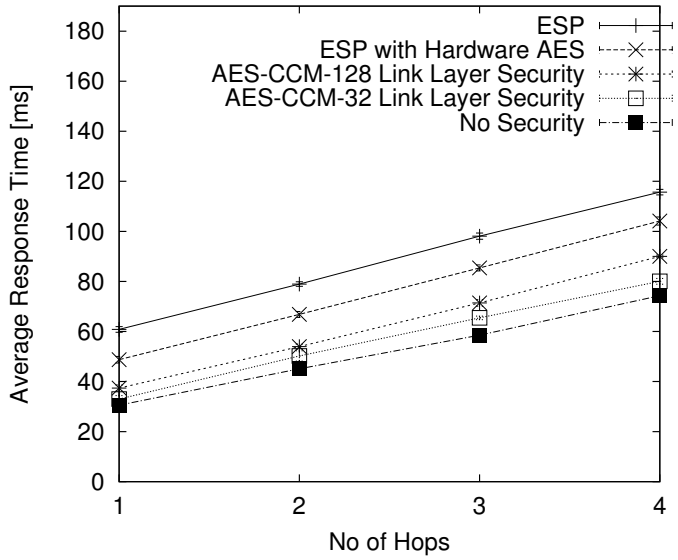


(a) Single hop with different data sizes

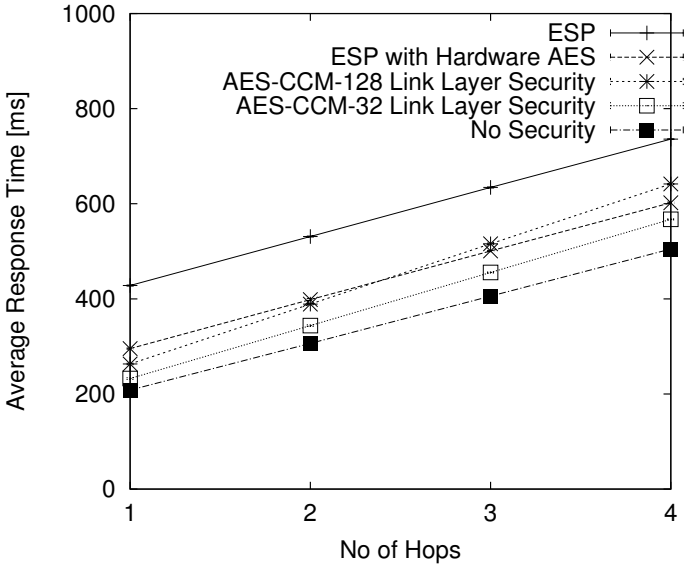


(b) Four hops with different data sizes

Figure 8.12: Response time versus datagram size when *integrity* is enabled. The 802.15.4 link-layer Security is better for single hop while IPsec gets better for multihop networks. The overhead of IPsec is constant across a single hop and a multihop for different data sizes while link-layer security overhead increases with the increasing data sizes.



(a) Multi hops with 16 bytes data size



(b) Multi hops with 512 byte data size

Figure 8.13: Response time versus number of hops when *integrity* and *confidentiality* is enabled. The 802.15.4 link-layer security is better for small data sizes while IPsec gets better for large data sizes across multiple hops.

comes faster than link-layer security. This result is opposed to the common belief that link-layer security is the fastest solution in all cases.

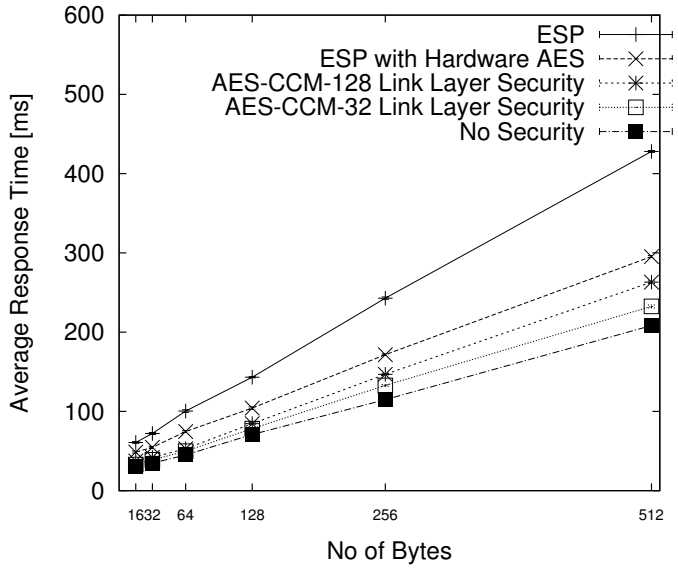
Summary

Our evaluation shows that IPsec fits in a tiny node (e.g. Tmote Sky) with room still available for applications. Our cryptographic algorithms analysis show that our implementation of AES-CTR and AES-XBC-MAC-96 are the fastest and most energy efficient. We showed that link-layer security is more efficient for the end-node in our experiment while IPsec is more efficient for forwarding nodes. In absolute terms, the energy overheads we measured are not significant when compared to the energy consumption of a node in steady-state operation. The system-wide response time shows a counter-intuitive result: in spite of its higher software complexity, IPsec provides a better scalability than link-layer security. By raising the level at which the security is guaranteed, it substantially reduces the data overhead on fragmented traffic. Thus, for large IP packets and/or a large number of hops, IPsec is faster than link-layer security. Our results also show that IPsec can be substantially accelerated with the help of the cryptographic co-processor included in modern radio chips such as the CC2420.

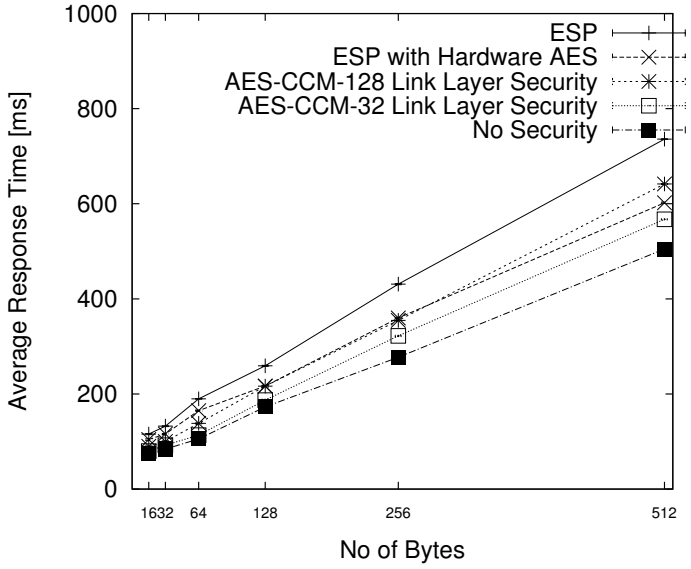
8.7 Conclusion

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

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



(a) Single hop with different data sizes



(b) Four hops with different data sizes

Figure 8.14: Response time versus datagram size when *integrity* and *confidentiality* is enabled. The 802.15.4 link-layer Security is better for single hop while IPsec gets better for multihop networks. The overhead of IPsec is constant across a single hop and a multihop network for different data sizes while link-layer security overhead increases with the increasing data sizes.

Acknowledgments

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

Bibliography

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

[2] *IEEE std. 802.15.4 - 2003: Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications for Low Rate Wireless Personal Area Networks (LR-WPANs)*. IEEE, 2003.

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

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

[5] A. Dunkels, B. Grönvall, and T. Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *1st IEEE Workshop on Embedded Networked Sensors (EmNetS'04)*, Tampa, USA, 2004.

[6] A. Liu and P. Ning. TinyECC: A configurable library for elliptic curve cryptography in wireless sensor networks. In *7th International Conference on Information Processing in Sensor Networks (IPSN'08)*, Washington, DC, USA, 2008.

[7] P. Szczechowiak, L. Oliveira, M. Scott, M. Collier, and R. Dahab. Nanoecc: Testing the limits of elliptic curve cryptography in sensor networks. In *5th European conference on Wireless Sensor Networks (EWSN'08)*, Bologna, Italy, 2008.

[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 *4th ACM Conference on Networked Embedded Sensor Systems (SenSys'06)*, Boulder, USA, 2006.

[9] W. Hu, P. Corke, W. Shih, and L. Overs. secfleck: A public key technology platform for wireless sensor networks. In *6th European conference on Wireless Sensor Networks (EWSN'09)*, Cork, Ireland, 2009.

[10] 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.

[11] A. Chung and U. Roedig. DHB-KEY: An Efficient Key Distribution Scheme for Wireless Sensor Networks. In *4th IEEE International Workshop on Wireless and Sensor Networks Security (WSNS'08)*, Atlanta, USA, 2008.

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

[13] R. Moskowitz. HIP Diet EXchange (DEX). draft-moskowitz-hip-rg-dex-05, 2011.

[14] Rodrigo Roman, Cristina Alcaraz, Javier Lopez, and Nicolas Sklavos. Key management systems for sensor networks in the context of the internet of things. *Computers and Electrical Engineering*, 37(2):147 – 159, 2011.

[15] T. Dierks and C. Allen. The tls protocol version 1.0. RFC 2246, 1999.

[16] S. Hong, D. Kim, M. Ha, S. Bae, S. J. Park, W. Jung, and J. Kim. Snail: an ip-based wireless sensor network approach to the internet of things. *Wireless Communications, IEEE*, 17(6):34–42, 2010.

[17] S. Fouladgar, B. Mainaud, K. Masmoudi, and H. Afifi. Tiny 3-tls: A trust delegation protocol for wireless sensor networks. In *3rd European Workshop on Security and Privacy in Ad-Hoc and Sensor Networks (ESAS'03)*, Hamburg, Germany, 2006.

[18] J. Granjal, R. Silva, E. Monteiro, J. Sa Silva, and F. Boavida. Why is IPsec a viable option for wireless sensor networks . In *4th IEEE International Workshop on Wireless and Sensor Networks Security (WSNS'08)*, Atlanta, USA, 2008.

[19] R. Riaz, Ki-Hyung Kim, and H.F. Ahmed. Security analysis survey and framework design for ip connected lowpans. In *9th International Symposium on Autonomous Decentralized Systems (ISADS'09)*, Athens, Greece, 2009.

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

[21] C. Alcaraz, P. Najera, J. Lopez, and R. Roman. Wireless sensor networks and the internet of things: Do we need a complete integration? In *1st International Workshop on the Security of the Internet of Things (SecIoT'10)*, Tokyo, Japan, 2010.

[22] Jorge Granjal, Edmundo Monteiro, and Jorge Sá Silva. Enabling network-layer security on ipv6 wireless sensor networks. In *IEEE Global Communications Conference (GLOBECOM,10)*, Miami, USA, 2010.

[23] Shahid Raza, Simon Duquennoy, Tony Chung, Dogan Yazar, Thiemo Voigt, and Utz Roedig. Securing Communication in 6LoWPAN with Compressed IPsec. In *Proceedings of the 7th IEEE International Conference on Distributed Computing in Sensor Systems (IEEE DCOSS 2011)*, Barcelona, Spain, June 2011.

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

[25] J. Hui and P. Thubert. Compression Format for IPv6 Datagrams in Low Power and Lossy Networks. draft-ietf-6lowpan-hc-15, 2011.

[26] Naveen Sastry and David Wagner. Security considerations for ieee 802.15.4 networks. In *Proceedings of the 3rd ACM workshop on Wireless security*, WiSe '04, pages 32–42, New York, NY, USA, 2004. ACM.

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

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

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

[30] S. Sakane, K. Kamada, M. Thomas, and J. Vilhuber. Kerberized Internet Negotiation of Keys (KINK). RFC 4430, 2006.

[31] C. Kaufman, P. Hoffman, Y. Nir, and P. Eronen. Internet Key Exchange Protocol Version 2 (IKEv2). RFC 5996 (Proposed Standard), 2010. Updated by RFC 5998.

[32] M. Richardson. A Method for Storing IPsec Keying Material in DNS. RFC 4025 (Proposed Standard), 2005.

[33] J. Polastre, R. Szewczyk, and D. Culler. Telos: enabling ultra-low power wireless research. In *4th International Conference on Information Processing in Sensor Networks (IPSN'05)*, Los Angeles, USA, 2005.

[34] Shamus Software. Multiprecision Integer and Rational Arithmetic C/C++ Library.

[35] N. Tsiftes and A. Dunkels. A database in every sensor. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Seattle, WA, USA, 2011.

[36] J. Eriksson, A. Dunkels, N. Finne, F. Österlind, and T. Voigt. Mspsim – an extensible simulator for msp430-equipped sensor boards. In *4th European conference on Wireless Sensor Networks (EWSN'07)*, Delft, The Netherlands, 2007. Demo Session.

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

[38] A. Dunkels, J. Eriksson, N. Finne, and N. Tsiftes. Powertrace: Network-level power profiling for low-power wireless networks. Technical Report T2011:05, Swedish Institute of Computer Science, 2011.

[39] C. Park, K. Lahiri, and A. Raghunathan. Battery discharge characteristics of wireless sensor nodes: An experimental analysis. In *Proceedings of the IEEE Conf. on Sensor and Ad-hoc Communications and Networks (SECON)*, Santa Clara, California, USA, September 2005.

Chapter 9

Paper D: Lithe: Lightweight Secure CoAP for the Internet of Things

Shahid Raza, Hossein Shafagh, Kasun Hewage, René Hummen, Thiemo Voigt.
In Submission, 2013.

Abstract

The Internet of Things (IoT) enables a wide range of application scenarios with potentially critical actuating and sensing tasks, e.g., in the e-health domain. For communication at the application layer, resource-constrained devices are expected to employ the Constrained Application Protocol (CoAP) that is currently being standardized at the IETF. To protect the transmission of sensitive information, secure CoAP (CoAPs) mandates the use of Datagram TLS (DTLS) as the underlying security protocol for authenticated and confidential communication. DTLS, however, was originally designed for comparably powerful devices that are interconnected via reliable, high-bandwidth links.

In this paper, we present Lithe– an integration of DTLS and CoAP for the IoT. With Lithe, we additionally propose a novel DTLS header compression scheme that aims to significantly reduce the header overhead of DTLS leveraging the 6LoWPAN standard. Most importantly, our proposed DTLS header compression scheme does not compromise the end-to-end security properties provided by DTLS. At the same time, it considerably reduces the number of transmitted bytes while maintaining DTLS standard compliance. We evaluate our approach based on a DTLS implementation for the Contiki operating system. Our evaluation results show significant gains in terms of packet size, energy consumption, processing time, and network-wide response times, when compressed DTLS is enabled.

9.1 Introduction

IPv6 over Low-power Wireless Personal Area Network (6LoWPAN) [1] enables the use of IP in low-power and lossy wireless networks such as Wireless Sensor Networks (WSNs). Such IP-connected smart devices (*Things*) are becoming part of the Internet hence forming the Internet of Things (IoT) or strictly speaking IP-connected IoT. TCP performance is known to be bad in wireless networks and the situation is exacerbated with the low power radios and lossy links found in sensor networks. Therefore, the connection-less UDP is mostly used in the IoT. Also HTTP, which is primarily designed to run over TCP, is inefficient in lossy and constrained environments. Therefore, a new connection-less lightweight Constrained Application Protocol (CoAP) [2] is being standardized for the IoT. Security is particularly important for the *Things* as they are connected to the untrusted Internet. Medical monitoring denotes a typical security-sensitive application scenario. Here, a smart device, such as an insulin pump[1], may be attached to the patient's body and periodically report the condition of the patient to a back-end service in the Internet. In emergency cases, a physician may additionally be able to trigger instant injection of medication into the patient's body.

CoAP proposes to use the Datagram Transport Layer Security (DTLS) [2] as the security protocol for automatic key management and for data encryption and integrity protection, as well as for authentication. CoAP with DTLS support is termed secure CoAP (CoAPs). While DTLS supports a wide range of cryptographic primitives for peer authentication and payload protection, it was originally designed for network scenarios where message length was not a critical design criterion. As such, DTLS is a chatty protocol and requires numerous message exchanges to establish a secure session. Therefore, it is inefficient to use the DTLS protocol, as it is, for constrained IoT devices. To cope with constrained resources and the size limitations of IEEE 802.15.4-based networks[2], 6LoWPAN header compression mechanisms are defined. The 6LoWPAN standard already defines the header compression format for the IP header, IP extension headers, and the UDP header. We believe, it is particularly advantageous to apply the 6LoWPAN header compression mechanism to compress other protocols having well-defined header fields, such as DTLS.

In this paper we provide lightweight CoAPs by compressing the underneath DTLS protocol [3] with 6LoWPAN header compression mechanisms. We call our lightweight 6LoWPAN compressed CoAPs *Lithe*. The purpose

[1] Medtronic probes insulin pump risks, Reuters, October 2011.

[2] The Maximum Transmission Unit (MTU) size of the IEEE 802.15.4 protocol is 127 bytes.

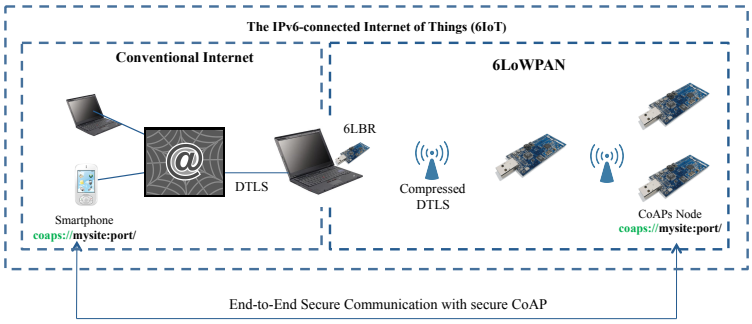


Figure 9.1: An IoT setup that uses CoAPs to secure communication between sensor nodes in 6LoWPANs and hosts in the Internet.

of DTLS header compression is twofold. First, to achieve energy efficiency reducing the message size, since communication requires more energy than computation. Second, to avoid 6LoWPAN fragmentation that is applied when the size of datagram is larger than the link layer MTU. Avoiding fragmentation, whenever possible, is also important from the security point of view as the 6LoWPAN protocol is vulnerable to fragmentation attacks [4]. Our compressed DTLS maintains true End-to-End (E2E) security between Lithe enabled hosts in 6LoWPAN networks and typical Internet hosts that use uncompressed CoAPs. Figure 9.1 shows a typical IoT setup, where a 6LoWPAN network consisting of CoAPs enabled nodes is connected through a 6LoWPAN Border Router (6BR) with the Internet. In this network setup, the CoAPs enabled devices can securely communicate with Internet hosts, such as standard computers, smartphones, etc., which support the CoAPs protocol.

To the best of our knowledge we are the first to propose 6LoWPAN compressed DTLS and enable lightweight CoAPs support for the IoT. We implement our DTLS header compression mechanisms in the Contiki OS [5], an open source and widely used operating system for constrained devices. We evaluate Lithe in an IoT setup and show the gains in terms of packet size reduction, energy consumption, processing time, and network-wide response time, without compromising security properties of DTLS and by maintaining standard compliant E2E security.
The main contributions of this paper are:

- We provide novel and standard compliant DTLS compression mechanisms that aim to increase the applicability of DTLS and, thus, CoAPs

for constrained devices.

- We implement the compressed DTLS in an OS for the IoT and evaluate it on real hardware; the results quantitatively show that Lithe is in many aspects more efficient compared to the uncompressed CoAP/DTLS.

The rest of the paper is organized as follows. We first give a brief overview of the technologies used in this paper in Section 9.2. In Section 9.3, we introduce our DTLS header compression mechanisms. Our implementation is outlined in Section 9.4. In Section 9.5, we describe our network setup and discuss the evaluation results. Finally, Section 9.6 summarizes the related work and Section 9.7 concludes this paper.

9.2 Background

Due to the heterogeneity in the IoT, it is challenging to connect resource-constrained devices in a secure and reliable way. Currently, different protocols such as CoAP [2], 6LoWPAN [6], IPv6 Routing Protocol (RPL) [7] for Low-power and Lossy Networks (LLNs) are being standardized by the Internet Engineering Task Force (IETF) to enable the IoT. The focus of this paper is to enable secure yet efficient communication among IoT devices that utilize the CoAP protocol. In this section we highlight the technologies involved in the development of the lightweight CoAPs, the HTTPs variant for the IoT.

9.2.1 CoAP and DTLS

CoAP is a web protocol that runs over the unreliable UDP protocol and is designed primarily for the IoT. CoAP is a subset of the most used synchronous web protocol HTTP and is tailored for constrained devices and machine-to-machine communication. However, while CoAP provides a REST interface similar to HTTP, it focuses on being more lightweight and cost-effective than its variant for today's Internet. To protect CoAP transmissions, Datagram TLS (DTLS) has been proposed as the primary security protocol by standardization [3]. Analogous to TLS-protected HTTP (HTTPs), the DTLS-secured CoAP protocol is termed CoAPs. As a basis for the discussion of our proposed DTLS compression mechanisms, we now give a brief overview of the DTLS protocol.

DTLS consists of two layers: the lower layer contains the Record protocol and the upper layer contains either of the three protocols namely Handshake,

Alert and ChangeCipherSpec, or application data. The ChangeCipherSpec is used during the handshake process to merely indicate that the Record protocol should protect the subsequent messages with the newly negotiated cipher suite and security keys. DTLS uses the Alert protocol to communicate the error messages between the DTLS peers. Figure 9.2 shows the structure of a DTLS message in a IP/UDP datagram.

The Record protocol [3] is a carrier for the upper layer protocols. The Record header contains among others `content type` and `fragment` fields. Based on the value in the `content type` the `fragment` field contains either the Handshake protocol, Alert protocol, ChangeCipherSpec protocol, or application data. The Record header is primarily responsible to cryptographically protect the upper layer protocols or application data once the handshake process is completed. The Record protocol's protection includes, confidentiality, integrity protection and authenticity.

The DTLS Record is a rather simple protocol whereas the Handshake protocol is a complex chatty process and contains numerous message exchanges in an asynchronous fashion. Figure 9.3 shows a full handshake process. The handshake messages, usually organized in flights, are used to negotiate security keys, cipher suites and compression methods. The scope of this paper is limited to the header compression only and not the cryptographic processing of Record and Handshake protocols. For details of the individual handshake messages we refer to TLS[8] and DTLS [3].

DTLS guarantees E2E security of different applications on a single machine by operating between the transport and application layers. A secure version of CoAP (CoAPs) that uses DTLS as the underlying security protocol is already being standardized. A web resource on an IoT device can then be accessed securely via CoAPs protocol as:
`coaps://myIPv6Address:port/MyResource.xml`

9.2.2 6LoWPAN

The 6LoWPAN standard [1] defines the header compression and fragmentation of IPv6 datagrams within IPv6-connected WSNs, also called 6LoWPAN networks. The compression mechanism consists of IP Header Compression (IPHC) and Next Header Compression (NHC). The IPHC encodings can compress the IPv6 header length to 2 bytes for a single hop network and 7 bytes in a multi-hop case. Among other encoding bits in the IPHC is the NH bit that, when set, indicates the next header is compressed using NHC. The NHC is used to encode the IPv6 extension headers and UDP header. The NHC encod-

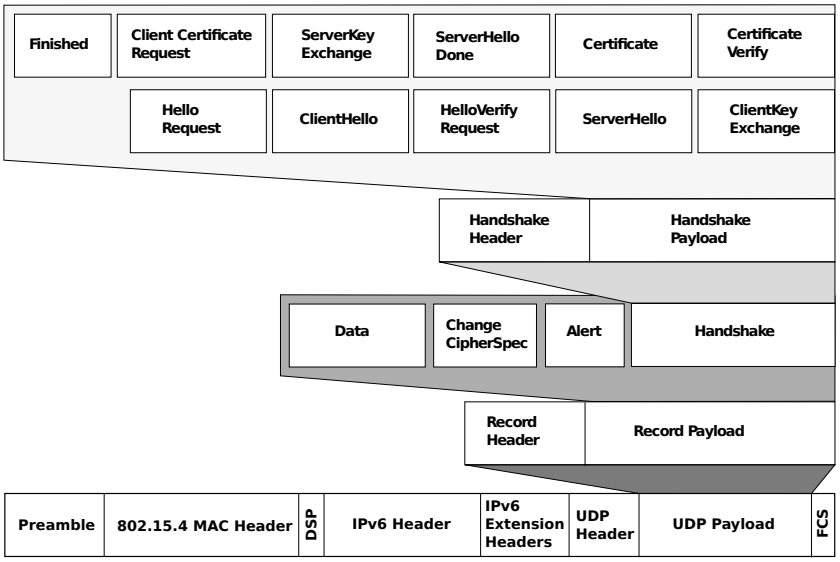


Figure 9.2: A layout of a packet secured with DTLS.

ings size is multiple of octets (mostly one byte) which contain variable length ID field and the encoding bits for a specific header. The 6LoWPAN standard defined NHC encoding can be used to compress headers up to UDP, and not the upper layer. This is because the NHC encodings for UDP do not contain NH bit that indicates that the UDP payload is also compressed. There are protocols that are part of UDP payload and have header-like structures similar to IP and UDP, such as DTLS, IKE. It is therefore worth applying the 6LoWPAN header compression mechanisms to compress these protocol headers. We provide 6LoWPAN NHCs to compress DTLS. Recently, Generic Header Compression (GHC) [9], analogous to NHC, is defined to allow upper layer (UDP payload and above) header compression. 6LoWPAN-GHC, a less flexible approach, is an alternative to our solution.

As depicted in Figure 9.1, the header compression is applied within the 6LoWPAN network only i.e. between constrained nodes and the 6LoWPAN border Router (6BR). A 6BR is used between 6LoWPAN networks and the Internet to compress/decompress or/and fragment/reassemble messages before routing to the two realms. In order to adapt chatty security protocols, such as DTLS, for the resource constrained IoT devices, it is wise to apply 6LoWPAN

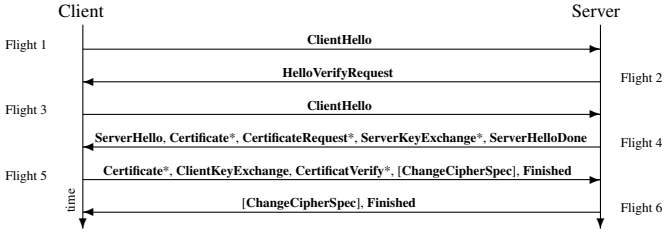


Figure 9.3: Full DTLS handshake protocol. Messages marked with a * are optional.

header compression mechanisms to these protocols as well. In Section 9.3 we propose 6LoWPAN header compression for DTLS. It is very important to design these header compression mechanisms in a way that complies with the plain DTLS standard, to be interoperable with existing and new DTLS enabled hosts on the conventional Internet.

The 6LoWPAN fragmentation part provides the schemes to fragment the datagrams when the packet size is bigger than the data-link layer Maximum Transmission Unit (MTU). The widely used data-link layer protocol in the 6LoWPAN networks is IEEE 802.15.4 that has MTU size of 127 bytes.

9.3 DTLS Compression

DTLS header compression, like IPHC, is applied only within 6LoWPAN networks, i.e., between sensor nodes and the 6BR. This is because DTLS is a part of the UDP payload and all information required for routing is already extracted at the IP layer. In this section, in addition to describing 6LoWPAN header compression for DTLS, we detail how our compressed DTLS can be linked to 6LoWPAN in a standard compliant way.

9.3.1 DTLS-6LoWPAN Integration

To cope with the size limitations of IEEE 802.15.4 link layer frames, we propose to compress the DTLS headers. The 6LoWPAN standard [1] does not provide ways to compress the UDP payload and the layers above. In order to apply 6LoWPAN header compression mechanisms to compress headers in the UDP payload, we either require a modification in the current NHC encoding for UDP in the 6LoWPAN standard, or need to define a new NHC for UDP

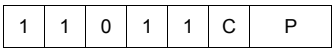


Figure 9.4: Our proposed 6LoWPAN-NHC for UDP, which allows compression of UDP payload.

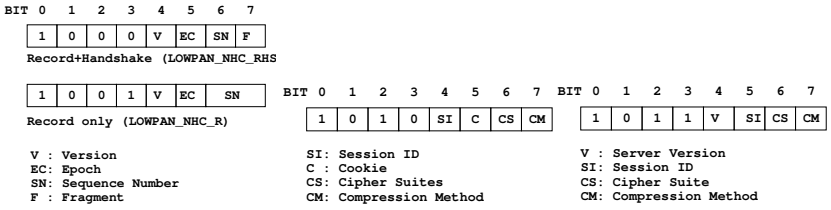
with different ID bits. The first solution requires modification is the current standard and hence not a favorable solution. The second solution is an extension to the 6LoWPAN standard; similar approach is adapted to distinguish NHC from GHC [9].

The ID bits 11110 in the NHC for UDP, as defined in the 6LoWPAN standard, indicate that the UDP payload is not compressed. We define ID bits 11011 to indicate that the UDP payload is compressed with 6LoWPAN NHC. These ID bits are currently unassigned in the 6LoWPAN standard. Figure 9.4 shows our proposed NHC for UDP that allows compression of UDP payload. In the following section we define 6LoWPAN-NHC for the DTLS Record header, Handshake header, and handshake messages where applicable.

9.3.2 6LoWPAN-NHC for the Record and Handshake Headers

The Record protocol adds 13 bytes long header fields to each packet that is sent throughout the lifetime of a device that uses DTLS/CoAPs. The Handshake protocol, on the other hand, adds 12 bytes of header to handshake messages. We propose 6LoWPAN-NHC for compressing the Record and Handshake headers, and reduce the header length to 5 and 3 bytes, respectively. In case of Handshake, only during the first handshake process the Handshake header and messages are compressed. This is because the successive re-handshake messages are encrypted using the negotiated cipher suite, and it is not possible to inspect payload of the DTLS record for compression at the 6LoWPAN layer. However, in all cases the Record header remains un-encrypted. Thus it is always compressed by using the mechanism explained in this section.

In order to provide header compression for the Record and Handshake header, we consider two cases. In the first case, where the Record header fragment field (see Section 9.2) contains a handshake message, we compress both the Record header and the Handshake header using a single encoding byte and we define 6LoWPAN-NHC for Record+Handshake (6LoWPAN-NHC-RHS). In the second case we define 6LoWPAN-NHC for the Record header (6LoWPAN-



(a) Record and the Handshake header.

(b) ClientHello message.

(c) ServerHello message.

Figure 9.5: Our proposed 6LoWPAN-NHC encodings for different DTLS headers.

NHC-R) where the fragment field in the Record header is application data and not a Handshake message as in the first case. The 6LoWPAN-NHC-R is applied after the DTLS handshake has been performed successfully, and the subsequently messages are encrypted and integrity protected. Figure 9.5a shows 6LoWPAN-NHC encodings for the Record+Handshake header and for the Record header. The encoded bits have the following functions:

The first four bits represent the ID field that is used to distinguish 6LoWPAN-NHC-RHS from other encodings, and to comply with 6LoWPAN-NHC encoding scheme. In case of 6LoWPAN-NHC-RHS we set the ID bits to 1000, and in case of 6LoWPAN-NHC-R we set the ID bits to 1001.

Version (V): If 0, the version is the DTLS latest version which is 1.2, and the field is omitted. If 1, the version field is carried inline.

Epoch (EC): If 0, an 8 bit epoch is used and the left most 8 bits are omitted. If 1, all 16 bits of the epoch are carried inline. In most cases the actual epoch is either 0 or 1. Therefore, an 8 bit epoch is used most of the time, allowing a higher compression ratio.

Sequence Number (SN): The sequence number consists of 48 bits, of which some are leading zeros. If SN is set to 0, a 16 bit sequence number is used and the left most 32 bits are omitted. If 1, all 48 bits of the sequence number are carried inline. In case of 6LoWPAN-NHC-R, as shown in Figure 9.5a, we use two bits for SN and can more efficiently compress the `sequence_number` field. Here if SN is set to 00, a 16 bit sequence number is used and the left most 32 bits are omitted. If 01, a 32 bit sequence number is used and the left most 16 bits are omitted. If 10, a 24 bit sequence number is used and the left most 24 bits are omitted. If 11, all 48 bits of the sequence number are carried

inline.
Fragment (F): If 0, the handshake message is not fragmented and the fields `fragment_offset` and `fragment_length` are omitted. This is the common case, which occurs when the handshake message is not larger than the maximum record size. If 1, the fields `fragment_offset` and `fragment_length` are carried inline.

In case of 6LoWPAN-NHC-R, `content_type` field is always carried inline, and in case of 6LoWPAN-NHC-RHS, `content_type` field is alway omitted as it is obvious from the NHC encoding that the content type is Handshake. Furthermore, `message_type` and `message_seq` fields of the Handshake header are always carried inline. The `length` field in the Record and Handshake headers is always omitted as they can be deduced from the lower layers: either from the 6LoWPAN header or the IEEE 802.15.4 header. We have to uncompress layer-wise from lower to higher layers until the the UDP header is uncompressed. Then the length of the UDP payload is known and the DTLS payload length can be calculated.

9.3.3 6LoWPAN-NHC for `ClientHello`

We propose 6LoWPAN-NHC for the `ClientHello` message (6LoWPAN-NHC-CH). During the handshake process the `ClientHello` message is sent twice, the first time without cookie and the second time with the server's cookie. Figure 9.5b shows 6LoWPAN-NHC encoding for the `ClientHello` message. The function of each compressed header field is described below:
The first four bits in the 6LoWPAN-NHC-CH represent the ID field which are set to 1010.
Session ID (SI): If 0, the `session_id` is not available and this field and 8 bits of the prefixed length field are omitted. In the (D)TLS protocol, `session_id` is empty if no session is available, or if the client wishes to generate new security parameters. The actual `session_id field` in the `ClientHello` contains 0 to 255 bits. However, it is always prefixed with an 8 bit field that contains the size of the `session_id`. The `ClientHello` message uses `session_id` only if the DTLS client wants to resume the old session. If SI is set 1, the `session_id` field is carried inline.
Cookie (C): If 0, the cookie field is not available and this field and its prefixed 8 bits length field are omitted. The actual cookie field in the ClientHello contains 0 to 255 bits[3]. However, it always has an 8 bits length field that contains the

[3]DTLS 1.2 specification increases the cookie size limit to 255 bytes; however, our implementation uses 255 bits.

size of the cookie. If C is set 1, the cookie field is carried inline.
Cipher Suite (CS): If 0, the default (mandatory) `cipher_suite` for CoAP that supports automatic key management is used and this field and the prefixed 16 bits length field are omitted. In the current CoAP draft [2]
`TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8`
is a mandatory cipher suite. The actual `cipher_suites field` contains 16 to $2^{16} - 16$ bits and is always prefixed with a 16 bits field that contains the size of the `cipher_suites`. If CS is set 1, the `cipher_suites` field is carried inline. *Compression Method (CM)*: If 0, the default compression method, i.e., `COMPRESSION_NULL` is used and this field and the prefixed 8 bits length field are omitted. The actual `compression_methods` field contains 8 to $2^8 - 8$ bits. It is always prefixed with an 8 bits field that contains the size of the `compression_methods`. If CM is set 1, the `compression_methods` field is carried inline.

The `random` field in the `ClientHello` is always carried inline whereas the version field is always omitted. The `version` contains the same value as in the DTLS Record header. In case of TLS/SSL the `version` field was defined to let a TLS client specify an older version to be compatible with an SSL client, which is rarely used in practice. All current versions of web browsers use the same TLS version in the Record header and in the `ClientHello` message. DTLS 1.2 (adapted from TLS 1.2) [8] mentions that the client sends its latest supported version in the `ClientHello`. All DTLS versions (1.0 and 1.2) have compatible `ClientHello` messages. If the server does not support this version, then the `ServerHello` message contains its supported version. If the client is not capable of handling the server's version, it terminates the connection with a protocol version alert.

Using 6LoWPAN-NHC-CH, usually only the `random` field in the `ClientHello` message is transmitted and all the other fields are omitted. The `ClientHello` with cookie may also contain the compressible `cookie` filed. Figure 9.6 shows an uncompressed IP/UDP datagram that contains a `ClientHello`. A 6LoWPAN compressed IP/UDP datagram, with our proposed compressed DTLS, containing the `ClientHello` message is depicted in Figure 9.7. After applying IPHC and 6LoWPAN-NHC header compression, the datagram size is significantly reduced.

9.3.4 6LoWPAN-NHC for `ServerHello`

We propose 6LoWPAN-NHC for the `ServerHello` message (6LoWPAN-NHC-SH). `ServerHello` is very similar to `ClientHello` except that the

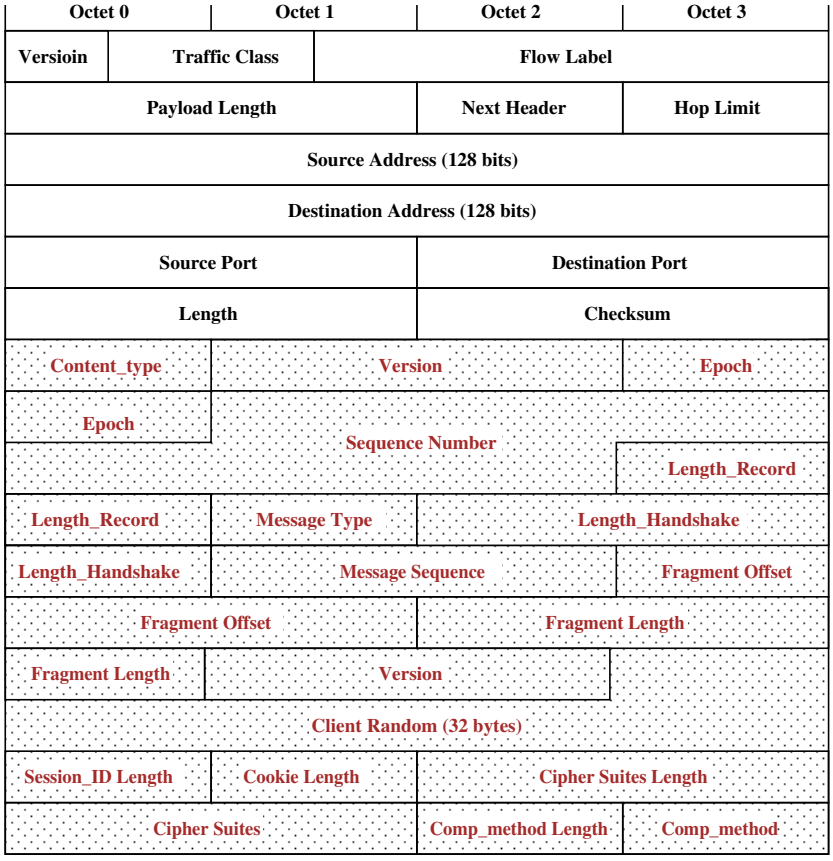


Figure 9.6: An *un-compressed* full IP/UDP datagram containing a DTLS `ClientHello` Message.

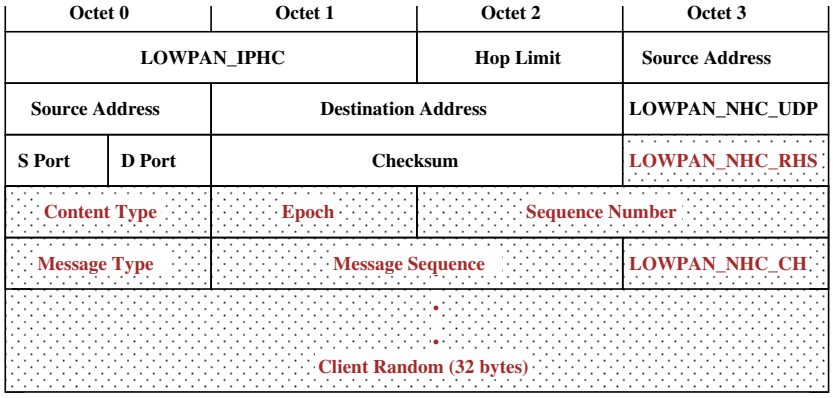


Figure 9.7: A 6LoWPAN *compressed* full IP/UDP datagram containing a DTLS `ClientHello` Message.

length of the `cipher_suites` and `compression_methods` fields are fixed to 16 and 8 bits, respectively. Figure 9.5c shows the 6LoWPAN-NHC encoding for the `ServerHello` message. The function of each compressed header field is described below:
The first four bits in the 6LoWPAN-NHC-SH represent the ID field set to 1011.
Version (V): In order to avoid version negotiation in the initial handshake, the DTLS 1.2 standard suggests that the server implementation should use DTLS version 1.0. If V is set to 0, the version is DTLS 1.0 and the version field is omitted. However the DTLS 1.2 clients must not assume that the server does not support higher versions or it will eventually negotiate DTLS 1.0 rather than DTLS 1.2 [3]. If V is set to 1, the version field is carried inline.
Session ID (SI), *Cipher Suite (CS)*, and *Compression Method (CM)* are encoded in a similar fashion as discussed in Section 9.3.3. In order to not compromise security the `random` field in the `ServerHello` is always carried inline.

9.3.5 6LoWPAN-NHC for other Handshake Messages

The remaining mandatory handshake messages `ClientKeyExchange`, `ServerHelloDone`, and `Finish` have no fields that could be compressed, hence all fields are carried inline. The optional handshake messages `Certificate` that contains the chain of certificates and `CertificateVerify` that contains the digital signature of the handshake message are as well carried inline.

However, it is possible to compress some of the fields inside a `Certificate` message which is out of the scope of this paper. Pritikin et al. propose a scheme to compress X.509 certificates [10].

The `ServerKeyExchange` message is mostly not sent, either due to crypto export restrictions or because the server's `Certificate` message contains enough information to concede the client to exchange the premaster secret. However, if it is sent, all fields are carried inline. In case of the optional message `CertificateRequest` all fields can be omitted. This is possible since the values for the fields `supported_signature_algorithms`, `certificate_types`, and `certificate_authorities` can be predefined to a single set of supported and preferred values for a 6LoWPAN network and all nodes in the network use the same set of values. The 6BR can populate the empty `CertificateRequest` message with the default set of values before sending the message to the destination in the conventional Internet. If no default set of values is defined for the 6LoWPAN network, all fields are carried inline.

9.4 Implementation

We implement Lithe in Contiki [5], an open source operating system for the IoT. However, our proposed header compression mechanisms in Lithe can be implemented in any OS that supports 6LoWPAN. The Lithe implementation consists of four main components: *(i)* DTLS, *(ii)* CoAP, *(*iii) integration module, *(iv)* DTLS header compression. For DTLS we use the open source tinyDTLS [11] implementation which supports the basic cipher suite based on preshared keys:
`TLS_PSK_WITH_AES_128_CCM_8`.
We adapt tinyDTLS for the WiSMote platform and for the 20-bit address support of msp430-gcc [12] (version of 4.7.0). For CoAP, we use the default CoAP implementation in the Contiki OS. We develop the integration module that connects the CoAP and DTLS implementations and enables the CoAPs protocol. This integration allows the application independent access to CoAPs where outgoing CoAP messages are transparently handed to DTLS that transmits the protected messages to the destination. All incoming CoAP messages are protected through DTLS and therefore are processed first at DTLS layer and handed transparently to CoAP, which resides in the application layer.

We implement our proposed header compression as an extension to the 6LoWPAN implementation in the Contiki OS. The 6LoWPAN layer resides

DTLS Header	Without Comp. [Bit]	With Comp. [Bit]	Comp. rate
Record	104	40[1]	62%
Handshake	96	24[1]	75%
`ClientHello`	336[2]	264[2]	23%
`ServerHello`	304	264[3]	14%
`CertificateRequest`	40	0	100%

Table 9.1: We send significantly *less* bits when the DTLS header compression is enabled.
[1] An additional byte is required to encode both the Record and Handshake headers.
[2] Some fields have a variable length. Here we only consider bits that are always sent.
[3] We do not compromise on security and send full size `random`. All other fields can be omitted.

between the IP and Medium Access Control (MAC) layers. The packets from the IP layer are considered as input packets and the packets from the MAC layer are considered as output packets. The 6LoWPAN layer processes all UDP packets from both directions. Therefore, we use two ways to distinguish UDP packets that carry DTLS messages as payload from other UDP packets. In the case of input packets, the pre-configured default DTLS port is used to identify CoAPs messages. In the second case when the packet is received from the MAC layer, the DTLS port and the ID bits in our proposed NHC-for-UDP and in the NHC for DTLS headers are used to distinguish the compressed headers from the uncompressed. Details are provided in Section 9.3.

Furthermore, it is important to emphasize, that while applying header compression, the E2E security of DTLS is by no means compromised. This is due to the design of DTLS and our effort to remain standard-compliant. The header fields are, after final negotiation of the cipher suite, integrity protected within the Record layer. During the compression/decompression process the original headers are not modified and the integrity protection is maintained. After decompression in the 6LoWPAN layer, the packets integrity is checked in the DTLS layer. The correctness of integrity protection serves as well a proof of correct decompression.

9.5 Evaluation

We evaluate Lithe on real sensor nodes running the Contiki OS. We use WiSMote [13] as our hardware platform. WiSMotes are equipped with *(i)* a 16 MHz, MSP430 5-Series, 16-bit RISC microcontroller, *(ii)* 16kB of RAM, *(iii)*

128kB of ROM, and *(iv)* an IEEE 802.15.4 (CC2520) transceiver. We select WiSMotes because of the RAM and ROM requirements of the DTLS implementation, which is discussed in more details in Section 9.5.2. The network setup consists of two WiSMotes which communicate directly through the radio. The CC2520 transceiver provides an AES-128 security module. However, for our evaluation we do not use the AES hardware support and rely on software AES. Leveraging the AES hardware support for the cryptographic computations involved in DTLS would lead to higher performance. The focus of our evaluation is on the impact of DTLS header compression on response time and energy consumption of nodes. Therefore, the performance loss due to software AES is not affecting our evaluation. Furthermore, we do not enable link layer security support, in order to be able to analyze the compression gain separately. In our previous work [14], we have evaluated the performance gains when using the AES support in hardware. There, we implement and evaluate the 802.15.4 link layer security.

9.5.1 Packet Size Reduction

Using 6LoWPAN-NHC compression mechanisms we can significantly reduce the length of DTLS headers. Table 9.1 shows that our proposed DTLS header compression significantly reduces the number of header bits which results in a similar reduction of radio transmission time.

The Record header, included in all DTLS messages, can be compressed by 64 bits (62%) for each message. In the case of the Handshake header, a compression rate of 75% is achieved. Application data constitutes the highest amount of DTLS message. Reducing the Record header from 104 to 40 bits, allows for transmission of 64 bits more payload per packet. Packets that are larger than the link layer MTU are fragmented. Fragmentation does not only introduce more burden to the node and the network, it also brings security vulnerabilities [4] along. Therefore, it is preferable to avoid fragmentation, whenever possible. Using compression we avoid fragmentation or decrease the number of fragments when the payload is slightly above the fragmentation threshold. Furthermore, reducing the transmitted bits in constrained networks has a huge impact on the performance and lifetime of the network. Radio communication has in general about 10 times higher energy consumption than in-node computations [13]. The tradeoff with compression is additional in-node computation overhead for compression/decompression, and in return reducing the radio transmissions. The impact of this tradeoff is discussed in more detail in Section 9.5.3.

Feature	ROM [Byte]	RAM [Byte]
DTLS Crypto (SHA-256, CCM, AES)	6590	2868
DTLS	10662	989
Contiki OS	32145	4979
CoAP	8632	582
DTLS Compression	2820	1
Total	60849	9419

Table 9.2: ROM and RAM requirements for Lithe.

9.5.2 RAM and ROM Requirement

We analyze RAM and ROM with the *msp430-size* and *msp430- objdump* tools in the MSP430 toolchain. As depicted in Table 9.2, in total 60kB of ROM and 9.2kB of stationary RAM are required for Lithe.

The DTLS implementation including the cryptographic functionalities and the DTLS state-machine requires 16.8kB of ROM and 3.7kB of stationary RAM. This makes DTLS to the major contributor of ROM after the OS. The CoAP-Server requires 8kB of ROM and 0.5kB of stationary RAM. Our CoAP-server provides a single resource, that upon a CoAP *GET* request, sends back a response message with variable payload lengths. This is used in our evaluation to analyze the effect of compression on CoAPs messages with different payload lengths. The footprint of the CoAP depends on the offered resources. The implementation of our DTLS header compression mechanism requires only 2820B of ROM and 1B of RAM. The total ROM used by 6LoWPAN in Contiki for compression and fragmentation (without DTLS compression) is 3782B. This verifies that the compressed DTLS uses the same order of ROM as standard 6LoWPAN. Today's sensor nodes, such as WiSMote, with 128kB of ROM can surely accommodate compressed CoAPs along with other operating system components, and still offer significant space to applications.

9.5.3 Run-time Performance

We look at the run-time performance gains that we achieve when compressed DTLS is used and compare it with uncompressed DTLS. We perform these experiments in a 6LoWPAN network with enabled Radio Duty Cycling (RDC) and respectively with no RDC. When RDC is used, the radio is off most of the time and is turned on either in certain intervals to check the medium for in-

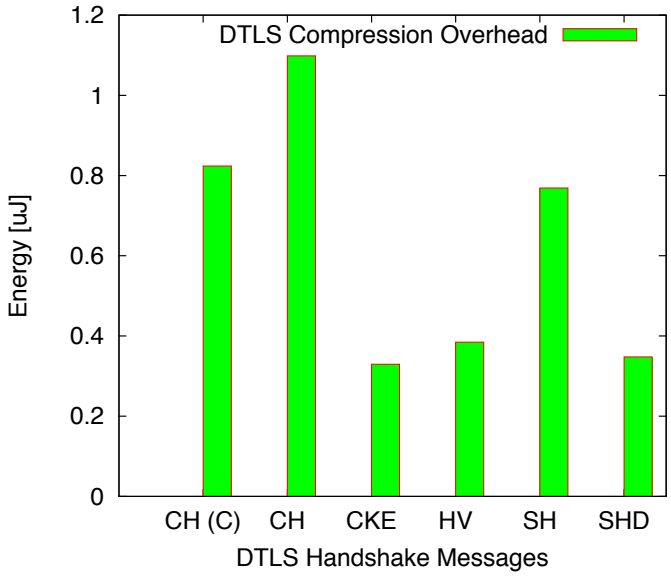


Figure 9.8: Energy consumption of individual compressed DTLS messages: ClientHello (CH), ClientKeyExchange (CKE), HelloVerify (HV), ServerHello (SH), ServerHelloDone (SHD)

coming packets or to transmit packets. We use the duty cycled MAC protocol, X-MAC [15] with its default settings, provided in the Contiki OS. In our runtime performance evaluation we focus on sensor node's energy consumption and network-wide round trip time. For the evaluation of energy consumption we use the energy estimation module [16] provided by Contiki OS. This module provides the usage time of CPU, LPM, Transmitter and Transceiver for a certain function call. The absolute timer values for each of these components can be converted to energy with the following equation:

$$\text{Energy } [mJ] = \frac{\text{ticks } \times \text{ I } [mA] \times \text{ Voltage } [V]}{\text{ticks per second}} \tag{9.1}$$

DTLS Compression Overhead

The overhead caused through in-node computation for compression and decompression of DTLS headers is almost negligible. However, we measure

Compression	Client-side [uJ]	Server-side [uJ]	Total [uJ]
Without	1756.66	1311.65	3068.31
With	1467.54	1143.47	2611.01

Table 9.3: Average energy consumption for packet transmission during DTLS handshake for the PSK cipher suite with no RDC. In average 15% energy saving for the transmission is achieved by compression.

and show it for the sake of completeness. Figure 9.8 shows the additional overhead for compression/decompression for the handshake messages. Each handshake message consists of the both Record and Handshake headers. For a DTLS handshake based on pre-shared keys, on average, an overhead of 4.2 uJ is caused with compression.

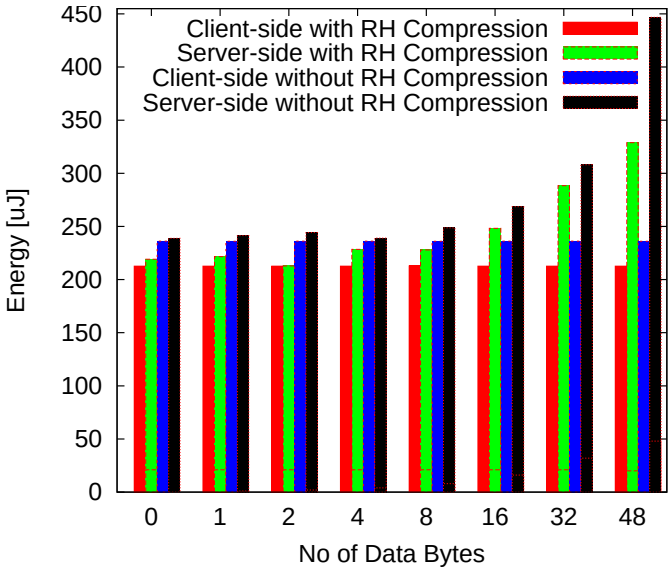
CoAPs Initialization

During the CoAPs initialization phase a secure session is established between the two communicating end points using the DTLS handshake protocol. The handshake process uses both the Record and Handshake headers, which means that both of these headers can be compressed. The tradeoff between additional in-node computation vs. reduced packet sizes shows itself in the energy consumption for packet compression in a DTLS handshake. Table 9.3 compares the energy consumption for overall transmission for the case compression is applied and respectively for the case transmission is not applied. On average 15% less energy is used to transmit (and receive) packets with compression. This is due to smaller packets sizes achieved through compression.

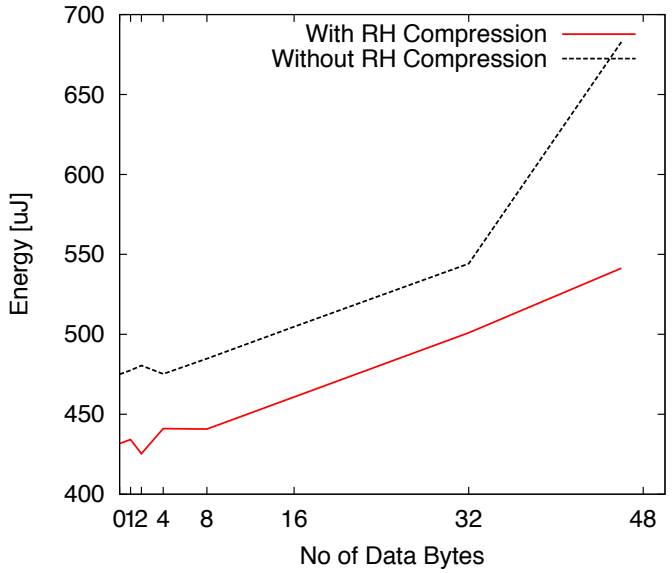
CoAPs Request-Response

Once the CoAPs initialization phase is completed, i.e. the handshake has been performed, a sensor node can send/receive secure CoAP messages using the DTLS Record protocol. Although the Handshake protocol is, compared to the Record protocol, a more resource hungry protocol, it is performed only once during the initialization phase and/or later (rarely) for re-handshake.

In order to measure the performance of compression within Record Header, we measure the energy consumption and the round trip time (RTT) for the processing of CoAP request-response messages. We start our measurements when the client prepares the CoAP request, and stop after the server's response is



(a) Energy consumed by client and server on transmission while sending compressed and uncompressed CoAPs messages of different data sizes



(b) Combined energy consumed by client and server on transmission while sending compressed and uncompressed CoAPs messages of different data sizes

Figure 9.9: The energy consumption of CoAPs messages when radio duty cycling is off shows that the compressed CoAPs message consumes less energy; the difference is significant when the messages are fragmented at the 6LoWPAN layer

received and processed. The corresponding CoAP response contains varying payload lengths. To be more precise, eight different payload sizes in the range of 0 to 48 bytes are used. We select 48 bytes, because with 48 byte CoAP payload 6LoWPAN fragmentation is performed in case of plain CoAPs. However, Lithe does not trigger fragmentation, due to the reduced bits by the means of compression. This effect is visible in Figure 9.9a, which shows the average in-node energy consumption on CoAPs' client and server for transmitting compressed and uncompressed CoAPs request and response pairs of different sizes with no RDC. The transmission of CaAP *GET* requests have the same amount of energy consumption since the size of request messages are always constant. Hence, energy consumption for CoAPs requests is always reduced by 10% using compression. The energy savings for the CoAPs response messages depend on the payload length and whether compression can prevent fragmentation. The latter is the case for a payload length of 48 Byte. Hence, the energy saving is in the range of 4-26%, where the highest energy saving is for 48 Byte.

For analyzing the overall energy consumption savings for CoAP request-responses, we sum up energy consumption for packet transmission on the server and the client as depicted in Figure 9.9b. We observe that in average energy savings of about 7% are achieved. However, in the case where fragmentation is avoided through compression, the savings increase to 20.6%. This is due the fact, that with 48 Byte payload, 6LoWPAN transmits the packet within two fragments, whereas with compression the packet is transmitted without fragmentation.

The reduced transmission time affects as well the RTT for a CoAP request-response message. In case of no RDC, as shown in Figure 9.10b, the RTT is in average 1.5% smaller, except for 48 Byte payload. There, the RTT with compression is even 77% smaller, since fragmentation is avoided. In order to assess the overall overhead caused through security, we have as well added values for CoAP without security. The RTT in CoAP without security is in average 1/3 of the CoAPs, as long as no fragmentation due to CoAPs is caused. Looking at the RTT with RDC, as shown in Figure 9.10b we see that for all three cases of: *a)* CoAP without any security, *b)* plain CoAPs, and *c)* CoAPs with DTLS compression (Lithe), RTT values are in the same range, expect for CoAP response messages with 48 Byte payload. This is a side-effect of RDC. RDC saves energy, by putting the radio in sleep for the most of the time. However, this happens at the cost of higher latency. Packets in RDC networks are not transmitted directly. The sender has to wait until the receiver wakes up and in the worst case this might be the whole sleeping interval of the receiver. As a result, the overall RTT is higher than when no RDC is used. We observe

that in networks with RDC, in the case compression prevents fragmentation or decreases the number of fragments, the RTT is significantly reduced. For example, in Figure 9.10b for 48 bytes Payload, compression leads to 50% shorter RTT.

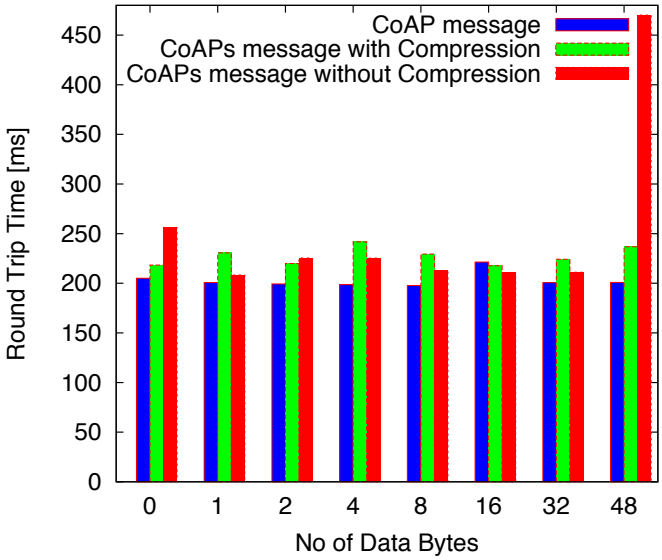
9.6 Related Work

Providing E2E security is a widely explored area in conventional Internet communication. However, there has been comparatively less research conducted in E2E security considering 6LoWPANs. The resource constraints of the devices and the lossy nature of wireless links are among the major reasons that hinder to apply general E2E security mechanisms to 6LoWPANs. Recently, the community has presented works on analyzing security challenges in the IP-based IoT [17] and solutions that improve or modify standard IP security protocols for the requirements of resource-constrained devices. In our discussion of related work, we focus on approaches that aim to enable E2E security solutions in the IoT.

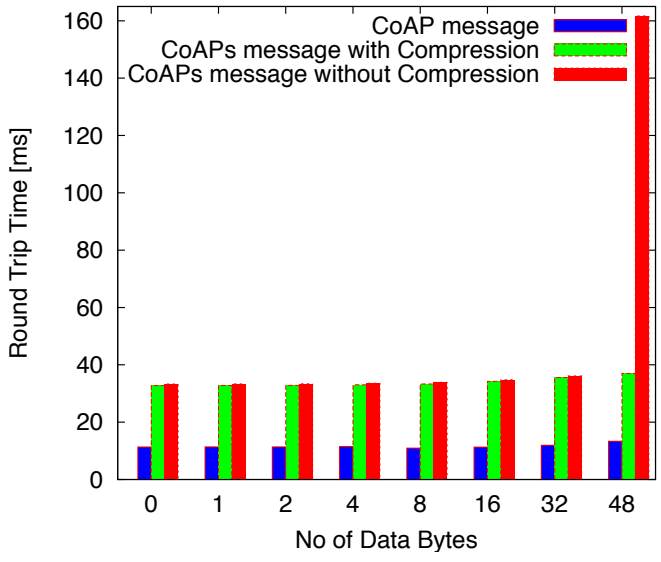
In our previous work [18], we propose a header compression method to use IPsec to secure the communication between nodes in 6LoWPAN networks and hosts in the Internet. We define NHC encodings to compress Authentication Header (AH) and Encapsulating Security Payload (ESP) extension headers. Jorge at al. [19] extend our solution and include IPsec in tunnel mode. They implement and evaluate their proposal in TinyOS. In [20], Brachmann et al. propose TLS-DTLS mapping to secure the IoT. However, this requires the presence of a trusted 6BR, and E2E security breaks at the 6BR.

Kothmayr et al. [21] investigate the use of DTLS in 6LoWPANs with a Trusted Platform Module (TPM) to get hardware support for the RSA algorithm. However, they have used DTLS as it is without using any compression method which would shorten the lifetime of the entire network due to the redundant bits in DTLS messages. Granjal et al. [22] evaluate the use of DTLS as it is with CoAP for secure communication. They note that payload space scarcity would be problematic with applications that require larger payloads. As an alternative, they suggest to employ security at other networking layers such as compressed form of IPsec.

In another work [23], we propose design ideas to reduce the energy consumption of the two-way certificate-based DTLS handshake. We suggest (i) pre-validation of certificates at the trusted 6BR, (ii) session resumption to avoid the overhead of a full handshake, and (iii) handshake delegation to the owner of



(a) With Radio Duty Cycling



(b) Without Radio Duty Cycling

Figure 9.10: Comparison of Round Trip Time for Lithe, plain CoAPs, and CoAP.

the resource-constrained device. That work in making certificate-based authentication viable for the IoT is complementary to this one. We plan to combine DTLS header compression with those ideas to make the mutual certificate-based handshake more efficient.

9.7 Conclusions

CoAP enabled hosts will be an integral part of the Internet of Things (IoT). Furthermore, real world deployments of CoAP supported devices require security solutions. To this end, DTLS is the standard protocol to enable secure CoAP (CoAPs). In this paper, we investigate if the overhead of DTLS can be reduced by 6LoWPAN header compression, and present the first DTLS header compression specification for 6LoWPAN. We quantitatively show that DTLS can be compressed and its overhead can be significantly reduced using 6LoWPAN standardized mechanisms. Our implementation and evaluation of compressed DTLS demonstrate that it is possible to reduce the CoAPs overhead as the DTLS compression is efficient in terms of energy consumption and network-wide response time, when compared with plain CoAPs. The difference between compressed DTLS and plain DTLS is very significant, if the use of plain DTLS results in 6LoWPAN fragmentation.

As future work we plan to deploy Lithe in a real world IoT system with a real application scenario. Such an IoT consists of constrained devices, standard computers, and smartphones. A real world deployment helps us to throughly evaluate Lithe in an heterogeneous IoT, and ultimately demonstrate the use of Lithe in security sensitive applications.

Bibliography

[1] J. Hui and P. Thubert. Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks. RFC 6282 (Proposed Standard), September 2011.

[2] Z. Shelby, K. Hartke, C. Bormann, and B. Frank. Constrained Application Protocol (CoAP), December 2012.

[3] E. Rescorla and N. Modadugu. Datagram Transport Layer Security Version 1.2. RFC 6347 (Proposed Standard), January 2012.

[4] R. Hummen, J. Hiller, H. Wirtz, M. Henze, H. Shafagh, and K. Wehrle. 6LoWPAN Fragmentation Attacks and Mitigation Mechanisms. In *Proceedings of the 6th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*, April 2013.

[5] A. Dunkels. The Contiki Operating System. Web page: http://www.contiki-os.org/. Visited 2013-02-15.

[6] 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.

[7] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander. RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. RFC 6550, March 2012.

[8] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard), August 2008. Updated by RFCs 5746, 5878, 6176.

[9] C. Bormann. 6LoWPAN Generic Compression of Headers and Header-like Payloads. draft-bormann-6lowpan-ghc-04, March 2012.

[10] D. McGrew and M. Pritikin. The Compressed X.509 Certificate Format. draft-pritikin-comp-x509-00, May 2010.

[11] O. Bergmann and C. Bormann. tinyDTLS. Web page: http://tinydtls.sourceforge.net/. Visited 2013-02-15.

[12] Texas Instruments and Red Hat. mspgcc. http://sourceforge.net/projects/mspgcc/. Visited 2013-02-15.

[13] LCIS and Aragosystems. WiSMote Sensor Node. Web page: http://wismote.org/. Visited 2013-02-15.

[14] S. Raza, S. Duquennoy, J. Höglund, U. Roedig, and T. Voigt. Secure Communication for the Internet of Things - A Comparison of Link-Layer Security and IPsec for 6LoWPAN. *Security and Communication Networks, Wiley*, January 2012.

[15] M. Buettner, G. V. Yee, E. Anderson, and R. Han. X-MAC: a short preamble mac protocol for duty-cycled wireless sensor networks. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, SenSys '06, pages 307–320, New York, NY, USA, 2006. ACM.

[16] A. Dunkels, F. Osterlind, N. Tsiftes, and Z. He. Software-based on-line energy estimation for sensor nodes. In *Proceedings of the 4th workshop on Embedded networked sensors*, EmNets '07, pages 28–32, New York, NY, USA, 2007. ACM.

[17] T. Heer, O. Garcia-Morchon, R. Hummen, S. Keoh, S. S. Kumar, and K. Wehrle. Security Challenges in the IP-based Internet of Things. *Springer Wireless Personal Communications Journal*, 2011.

[18] S. Raza, S. Duquennoy, A. Chung, D. Yazar, T. Voigt, and U. Roedig. Securing communication in 6LoWPAN with compressed IPsec. In *7th International Conference on Distributed Computing in Sensor Systems (DCOSS'11)*, Barcelona, Spain, 2011.

[19] J. Granjal, E. Monteiro, and J. S. Silva. Network-layer security for the Internet of Things using TinyOS and BLIP. *International Journal of Communication Systems*, 2012.

[20] M. Brachmann, S. L. Keoh, O. G. Morchon, and S. S. Kumar. End-to-end transport security in the IP-Based Internet of Things. In *Computer Communications and Networks (ICCCN), 2012 21st International Conference on*, pages 1 –5, August 2012.

[21] T. Kothmayr, C. Schmitt, W. Hu, M. Brunig, and G. Carle. A DTLS based end-to-end security architecture for the Internet of Things with two-way authentication. In *Local Computer Networks Workshops, 2012 IEEE 37th Conference on*, pages 956–963. IEEE, 2012.

[22] J. Granjal, E. Monteiro, and J. S. Silva. On the feasibility of secure application-layer communications on the Web of Things. In *Local Computer Networks (LCN), 2012 IEEE 37th Conference on*, pages 228 –231, October 2012.

[23] R. Hummen, J. H. Ziegeldorf, H. Shafagh, S. Raza, and K. Wehrle. Making Certificate-based Authentication Viable for the Web of Things. In *Proceedings of the 2nd ACM Workshop on Hot Topics on Wireless Network Security and Privacy (HotWiSec)*, April 2013.

Chapter 10

Paper E: SVELTE: Real-time Intrusion Detection in the Internet of Things

Shahid Raza, Linus Wallgren, Thiemo Voigt.
Ad Hoc Networks Journal, Elsevier, 2013 [Accepted]

Abstract

In the Internet of Things (IoT), resource-constrained things are connected to the unreliable and untrusted Internet via IPv6 and 6LoWPAN networks. Even when they are secured with encryption and authentication, these things are exposed both to wireless attacks from inside the 6LoWPAN network and from the Internet. Since these attacks may succeed, Intrusion Detection Systems (IDS) are necessary. Currently, there are no IDSs that meet the requirements of the IPv6-connected IoT since the available approaches are either customized for Wireless Sensor Networks (WSN) or for the conventional Internet.

In this paper we design, implement, and evaluate a novel intrusion detection system for the IoT that we call SVELTE. In our implementation and evaluation we primarily target routing attacks such as spoofed or altered information, sinkhole, and selective-forwarding. However, our approach can be extended to detect other attacks. We implement SVELTE in the Contiki OS and thoroughly evaluate it. Our evaluation shows that in the simulated scenarios, SVELTE detects all malicious nodes that launch our implemented sinkhole and/or selective forwarding attacks. However, the true positive rate is not 100%, i.e., we have some false alarms during the detection of malicious nodes. Also, SVELTE's overhead is small enough to deploy it on constrained nodes with limited energy and memory capacity.

10.1 Introduction

With IPv6 over Low-power Wireless Personal Area Network (6LoWPAN) [1, 2] it is possible to connect resource constrained devices, such as sensor nodes, with the global Internet using the standardized compressed IPv6 protocol. These networks of resource constrained devices, also called 6LoWPAN networks, and the conventional Internet form the Internet of Things or strictly speaking the IP-connected Internet of Things (IoT). A 6LoWPAN Border Router (6BR) is an edge node that connects 6LoWPAN networks with the Internet. Due to the resource constrained nature of the devices or *things*, 6LoWPAN networks mostly use IEEE 802.15.4 as link and physical layer protocol.

Unlike typical wireless sensor networks (WSN), 6LoWPAN networks or IP-connected WSN are directly connected to the untrusted Internet and an attacker can get access to the resource-constrained *things* from anywhere on the Internet. This global access makes the *things* vulnerable to intrusions from the Internet in addition to the wireless attacks originating inside 6LoWPAN networks. Potential applications of the IoT are smart metering, home or building automation, smart cities, logistics monitoring and management, etc. These applications and services are usually charged and the revenue is based on data or services used. Hence, the confidentiality and integrity of the data and timely availability of services is very important.

Researchers have already investigated message security for the IoT using lightweight DTLS [3], IPsec [4], and IEEE 802.15.4 link-layer security [5]. Even with message security that enables encryption and authentication, networks are vulnerable to a number of attacks aimed to disrupt the network. Hence, an Intrusion Detection System (IDS) is necessary to detect intruders that are trying to disrupt the network.

The available IDSs for WSNs could be used in the IoT. However, most of these approaches are built on the assumptions that *(i)* there is no central management point and controller, *(ii)* there exists no message security, and *(iii)* nodes cannot be identified globally. The IoT has a novel architecture where the 6BR is assumed to be always accessible, end-to-end message security is a requirement [5], and sensor nodes are globally identified by an IP address. Besides these opportunistic features, an IDS for the IoT is still challenging since the *things (i)* are globally accessible, *(ii)* are resource constrained, *(iii)* are connected through lossy links, and *(iv)* use recent IoT protocols such as CoAP [6], RPL [7], or 6LoWPAN [2]. Therefore, it is worth investigating and providing an IDS for the IoT exploiting these opportunities and threats.

To this end, we design, implement, and evaluate a novel Intrusion Detection

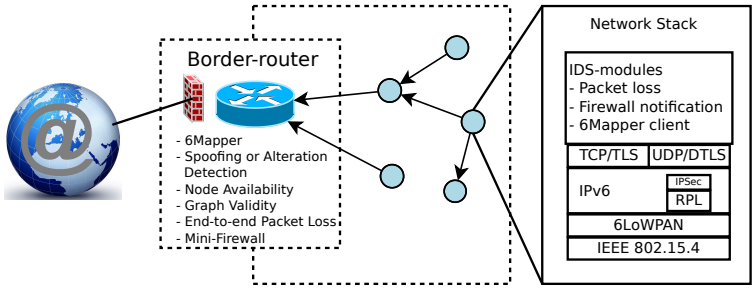


Figure 10.1: An IoT setup where IDS modules are placed in 6BR and also in individual nodes.

system for the IoT that we call SVELTE[1]. To the best of our knowledge this is the first attempt to develop an IDS specifically designed for the IoT. Network layer and routing attacks are the most common attacks in low power wireless networks [8], and in this paper we primarily target these attacks. SVELTE is also inherently protected against sybil and clone ID attacks; we discuss these attacks in Section 10.3.2. We evaluate SVELTE against sinkhole and selective-forwarding attacks. Our approach is, however, extensible and can be used to detect other attacks as we discuss in Section 10.7.

The IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) [7] is a novel standardized routing protocol primarily designed to meet the specific routing requirements of the IoT. SVELTE uses RPL as a routing protocol. It has two main components: the 6LoWPAN Mapper (6Mapper), and intrusion detection modules. The 6Mapper reconstructs RPL's current routing state, i.e., its directed acyclic graph, at the 6BR and extends it with additional intrusion detection parameters.

One of the important decisions in intrusion detection is the placement of the IDS in the network. We use a hybrid approach, see Section 10.3, and place the processing intensive SVELTE modules in the 6BR and the corresponding lightweight modules in the constrained nodes. Figure 10.1 presents an overview of our IDS that we explain in more detail in Section 10.3. One of our main design goals is that the IDS should be lightweight and comply with the processing capabilities of the constrained nodes.

In addition to the 6Mapper and the intrusion detection techniques, we also

[1] SVELTE literary means *elegantly slim*.

propose and implement a distributed mini-firewall to protect 6LoWPAN networks against global attackers from Internet. We implement SVELTE in the Contiki operating system [9].

The main contributions of this paper are:

- We present SVELTE, a novel IDS with an integrated mini-firewall for the IP-connected IoT that uses RPL as a routing protocol in 6LoWPAN networks.
- We implement SVELTE and thoroughly evaluate it for 6LoWPAN networks that consist of resource-constrained *things* and have lossy communication links.

The next section of this paper gives an overview of the technologies used in SVELTE. Section 10.3 describes SVELTE that includes 6Mapper, the actual intrusion detection techniques, and the firewall. In Section 10.4 we detail SVELTE's implementation for the Contiki OS. Section 10.5 presents our detailed performance evaluation of SVELTE. We highlight the current IDSs and their applicability in the IoT in Section 10.6. Section 10.7 discusses the possible extensions in SVELTE, and finally we conclude the paper in Section 10.8.

10.2 Background

In this section we briefly discuss the technologies involved in SVELTE for the IoT.

10.2.1 The Internet of Things

The Internet of Things (IoT) or strictly speaking IP-connected IoT is a hybrid network of tiny devices, typically WSNs, and the conventional Internet. Unlike typical WSN where devices are mostly resource constrained and unlike in the Internet where devices are mostly powerful, the nodes or *things* in the IoT are heterogeneous devices. An IoT device can be a light bulb, a microwave, an electricity meter, an automobile part, a smartphone, a PC or a laptop, a powerful server machine or a cloud, or potentially anything. Hence the number of potential devices that can be connected to the IoT are in hundreds of billion. IPv6's huge address space has been designed to address this issue.

To connect resource constrained nodes such as WSN with the Internet using IPv6, a compressed version of the IPv6 called 6LoWPAN has been standardized [1, 2]. The 6LoWPAN protocol enables the routing of IPv6 packets in the

IP-connected WSN (also called 6LoWPAN network) in a compressed and/or fragmented form. Compression is needed since 6LoWPAN's link and physical layer protocol, IEEE 802.15.4, has a Maximum Transmission Unit (MTU) of 127 bytes. 6LoWPAN supports multi-hop enabling nodes to forward packets on behalf of other nodes that are not directly connected to the 6LoWPAN border router (6BR). The 6BR is an end device that connects 6LoWPAN networks with the Internet.

A 6LoWPAN network is a multi-hop wireless network where communication links are usually lossy and devices are resource-constrained and often battery powered. Therefore, the connection-less User Datagram Protocol (UDP) is mostly used in 6LoWPAN networks. Further, connection oriented web protocols such as HTTP are not feasible and a new protocol, the Constrained Application Protocol (CoAP), is being standardized for the IoT. Further, a new routing protocol, IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) [7], is standardized. SVELTE is primarily designed for RPL-based 6LoWPAN networks.

10.2.2 RPL

The IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) [7] is a standardized routing protocol for the IP-connected IoT. RPL is a versatile protocol that enables many-to-one, one-to-many, and one-to-one communication. It creates a Destination-Oriented Directed Acyclic Graph (DODAG) and supports different modes of operation: uni-directional traffic to a DODAG root (typically the 6BR), and bi-directional traffic between constrained nodes and a DODAG root. A typical RPL DODAG is shown in Figure 10.2 where each node has a node ID (an IPv6 address), one or more parents (except for the DODAG root), and a list of neighbors. Nodes also have a rank that determines their individual position with respect to the DODAG root and relative to other nodes. Ranks strictly increase from the DODAG root to nodes and strictly decrease in the direction towards the DODAG root.

Every node in the RPL network must be able to determine whether packets are to be forwarded to its parents, i.e., upwards, or to its children. The most simple way for a node to accomplish this is to know all its descendants and use the route to its parent as default route for all other packets. In-network routing tables are required to separate packets heading upwards and the packets heading downwards in the network. This is the mechanism in the RPL implementation in the Contiki operating system [9], that we use in this paper to evaluate SVELTE.

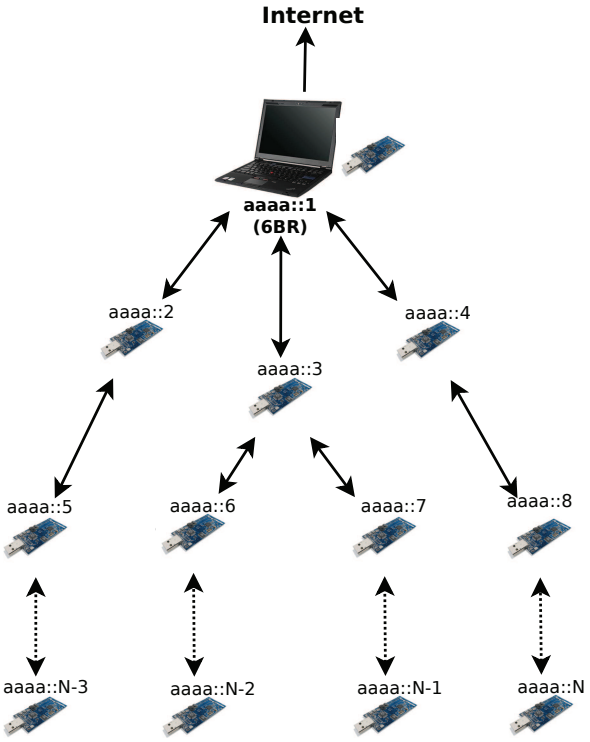


Figure 10.2: A sample RPL DODAG that has N nodes with IPv6 addresses from aaaa::1 to aaaa::N.

10.2.3 Security in the IoT

Real world IoT deployments require security. The communication between devices in the IoT can be protected on an end-to-end or on a hop-by-hop basis. IPsec [4] in transport mode provides end-to-end security between two hosts in the IoT. In IPv6 networks and hence in 6LoWPAN, IPsec is mandatory to implement, meaning that each IPv6-enabled device must have IPsec capabilities. IPsec's ESP protocol [10] ensures application data confidentiality and optionally data integrity and authentication, and AH [11] protocol ensures the integrity of whole IPv6 datagram that includes application data and IPv6 headers.

If the Constrained Application Protocol (CoAP) [6] is used in the IoT as

an application protocol then end-to-end security between two applications can be provided with the Datagram Transport Layer Security (DTLS). Also, IEEE 802.15.14 link-layer security can be used for per hop security.

Besides having message security, the IoT is vulnerable to a number of attacks [12] aimed to disrupt the network; hence intrusion detection mechanisms are important in real world IoT deployments, e.g., in building automation, industrial automation, smart metering and smart grids.

10.2.4 IDS

An Intrusion Detection System (IDS) is a tool or mechanism to detect attacks against a system or a network by analyzing the activity in the network or in the system itself. Once an attack is detected an IDS may log information about it and/or report an alarm. Broadly speaking, the detection mechanisms in an IDS are either signature based or anomaly based.

Signature based detections match the current behavior of the network against predefined attack patterns. Signatures are pre-configured and stored on the device and each signature matches a certain attack. In general signature based techniques are simpler to use. They need, however, a signature of each attack and must also store it. This requires specific knowledge of each attack and storage costs grow with the number of attacks. This approach is more static and cannot detect new attacks unless their signature is manually added into the IDS.

Anomaly based detection tries to detect anomalies in the system by determining the ordinary behavior and using it as baseline. Any deviations from that baseline is considered an anomaly. On one hand, anomaly based systems have the ability to detect almost any attack and adapt to new environments, but on the other hand these techniques have rather high false positive rates (to raise an alarm when there is no attack) as deviations from the baseline might be ordinary. Also, they have comparatively high false negative rates (no alarm when there is an attack) as attacks might only show a small deviation that is considered within the norm.

Keeping in view the novel requirements of the IoT, in this paper we use a hybrid of signature and anomaly based detections. We try to balance between the storage cost of the signature based detection and the computing cost of the anomaly based techniques. In SVELTE the detection techniques mostly target routing attacks such as sink-hole, selective forwarding, and spoofed or altered routing information [12]; however, SVELTE is extensible and can be used to detect other attacks as we discuss in Section 10.7.

In *sinkhole attacks* [12] an attacker advertises a beneficial routing path and thus makes many nodes route traffic through it. In RPL, an attacker can launch a sinkhole attack by advertising a better rank thus making nodes down in the DODAG select it as parent. In *selective forwarding attacks* [12], an attacker forwards only selected packets. For example, an attacker could forward only routing messages and drop all other packets to disrupt part of the network.

Once an attack is detected, the goal is to mitigate its effect and remove the attacker from the network. The simplest approach to remove an attacker is to ignore it. This requires the identification of the attacking node. Neither MAC nor IP addresses are trustworthy as they can be easily spoofed. One possible way to ignore a node is to use either a blacklist or a whitelist. A blacklist would include all malicious nodes and a whitelist would include all valid nodes. Maintaining a whitelist is easier in the presence of many attackers. In either way it is necessary that an attacker cannot obtain another valid identity, with sybil or clone ID attacks [12], as otherwise the attacker could restart the attack without effort. In SVELTE we use a whitelist.

10.3 SVELTE: An IDS for the IoT

Recall that a 6LoWPAN network is a lossy and wireless network of resource constrained nodes which uses IPv6 as networking protocol and often RPL as a routing protocol. One of the design goals of any protocol for the IoT is its ability to be deployed and run on constrained nodes in 6LoWPAN networks. Based on the novel requirements of the IoT, we propose SVELTE: a lightweight yet effective intrusion detection system for the IoT. We also compliment SVELTE with a distributed mini-firewall in order to filter malicious traffic before it reaches the resource constrained nodes.

We design SVELTE for a 6LoWPAN network that uses message security technologies, such as IPsec [4] and DTLS [3] to provide end-to-end message security. In the rest of this section we present our intrusion detection system.

Placement of SVELTE The placement of an IDS is an important decision that reflects the design of an IDS and the detection approaches. Keeping in view the resource constrained nature of the devices and the IoT setup shown in Figure 10.1, we use a hybrid, centralized and distributed, approach and place IDS modules both in the 6BR and in constrained nodes.

SVELTE has three main centralized modules that we place in the 6BR. The first module, called 6LoWPAN Mapper (6Mapper), gathers information about

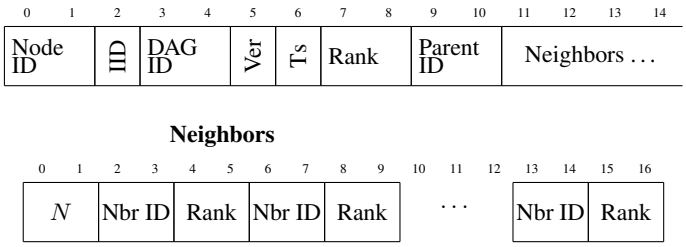


Figure 10.3: Packet format of the Mapping Response

the RPL network and reconstructs the network in the 6BR, as we describe in Section 10.3.1. The second module is the intrusion detection component that analyzes the mapped data and detects intrusion; Section 10.3.2 discusses this. The third module, a distributed mini-firewall, is designed to offload nodes by filtering unwanted traffic before it enters the resource constrained network; Section 10.3.3 details this. The centralized modules have two corresponding lightweight modules in each constrained node. The first module provides mapping information to the 6BR so it can perform intrusion detection. The second module works with the centralized firewall. Each constrained node also has a third module to handle end-to-end packet loss; this is discussed in Section 10.3.2.

10.3.1 6LoWPAN Mapper

A vital component of SVELTE is the 6LoWPAN Mapper (6Mapper) that reconstructs the RPL DODAG in the 6BR and complements it with each node's neighbor and parent information. To reconstruct the DODAG, the 6Mapper sends mapping requests to nodes in the 6LoWPAN network at regular intervals. The request packet contains the information necessary to identify an RPL DODAG. It includes the RPL Instance ID (IID), the DODAG ID, and the DODAG Version Number [7]. It also includes a timestamp (*Ts*) to know the recency of the mapping information received. The total size of a mapping request packet is 5 bytes.

Each node responds to the mapping request by perpending a Node ID to the request packet and by appending node rank, parent ID, and all neighbor IDs and ranks. An illustration of the mapping response packet format is shown in Figure 10.3. The basic response packet is 13 bytes long and requires an additional four bytes for each neighbor.

6Mapper with Authentic and Reliable Communication It is likely that IPsec Authentication Header (AH) [4] or IEEE 802.15.4 link-layer security are enabled in the IoT to protect the integrity of the IP headers. In this case there is no need to include the node ID in the response packet, as that would be the source address in the IP header. When the 6Mapper host, i.e., 6BR, has the same IPv6 address as the DODAG root it is also unnecessary to include the DODAGID that corresponds to the destination IP in the IP header. In the request packet the source and destination fields in the IP header have the opposite meaning, i.e., the IP source corresponds to the DODAGID and the destination corresponds to the node ID.

If mapping-packets are transferred reliably, for example, by using CoAP that employs acknowledgements, there is no need to send a timestamp with the mapping data as we can be sure that the packets arrive within the timeout specified for the underlying protocol. When the communication in the 6LoWPAN is authentic and reliable, the size of the 6Mapper request and response packets is reduced to 1 byte and 8 bytes, respectively.

Unidirectional RPL 6Mapper Some RPL implementations only support traffic destined to the DODAG root, typically the 6BR. To provide network mapping for these 6LoWPAN networks it is possible to alter the 6Mapper and let it wait for the periodic mapping response packets from each node without sending the explicit request packet. This solution has the additional advantage that it reduces traffic in the network which reduces power consumption. However, slightly more logic has to be added in each node which increases the memory consumption.

Valid inconsistencies in 6Mapper In our 6Mapper there is a possibility that mapping responses are inconsistent with each other, which can lead to false positives if not handled properly. This can happen if the information a node sends to the network mapper has become outdated or when an attacker deliberately changes the information. Below we show how valid routing graph inconsistencies occur. Consider a RPL DODAG where Node P is the parent of node C, the function $R_a(Node)$ represents the actual rank of $Node$ and $R_m(Node)$ represents the rank known to the 6Mapper.
- Node P sends its rank to the 6Mapper, $R_a(P) = 1024$ and $R_m(P) = 1024$
- Node P recalculates its rank and advertises it,
$R_a(P) = 512$ and $R_m(P) = 1024$
- Node C receives the updated rank from P

- Node C recalculates its rank. $R_a(C) = 768$
- Node C sends its rank to the 6Mapper, $R_a(C) = 768$ and $R_m(C) = 768$

As can be seen the state of the network is:
$R_a(P) = 512$
$R_a(C) = 768$
$R_m(P) = 1024$
$R_m(C) = 768$
This state is perfectly valid as node P has a better rank than node C, $R_a(P) < R_a(C)$. However, the 6Mapper assumes that the child, node C, has a better rank than its parent, which is inconsistent as $R_m(P) > R_m(C)$.

This is a problem which needs to be taken into consideration when designing methods for analyzing the mapped data. Leveraging the amount of sensors in a 6LoWPAN we improve the accuracy when faced with both natural and artificial inconsistencies; Section 10.3.2 discusses methods to overcome such inconsistencies.

Mapping requirements For our 6Mapper to be fully effective the packets used to map the network need to be indistinguishable from other packets. If an adversary can distinguish the traffic used by the 6Mapper from other traffic it is possible for an adversary to perform selective forwarding and only forward traffic necessary for the mapper, while dropping other traffic.

The first step to prevent this is to encrypt the data, to avoid that the packet content is revealed to an eavesdropping adversary. As mentioned earlier we assume that the message contents are protected with upper-layer security protocols such as IPsec or DTLS. Secondly, headers should not reveal any information that enables an eavesdropper to determine that the packet is used by the 6Mapper. Therefore it can be problematic if the source of the 6Mapper is the same for all nodes, as the IP header must be readable for all nodes. The adversary could use the IP header and the knowledge about the 6Mapper's host address to identify network mapping traffic. A simple solution to prevent this is to assign as many IPv6 addresses to the 6Mapper as there are nodes in the network. This is possible for RPL as IPv6 has a potentially unlimited address space of 2^{128} addresses. Thus, when an adversary compromises a node it will only know the node's mapping address and no other mapping addresses. Hence, it is not able to distinguish between ordinary traffic and mapping traffic for other nodes.

However, if the attacked node has more resources it may use more advanced traffic patterns and node behavior analysis techniques, and it might still be

possible for an adversary to distinguish between ordinary and mapper-related traffic.

10.3.2 Intrusion Detection in SVELTE

We design and implement three detection techniques which use the 6Mapper. The detection techniques primarily detect spoofed or altered information, sinkhole, and selective forwarding attacks. However, our approach is extensible and more attacks can be detected; we discuss some of the possible extensions in Section 10.7.

Network Graph Inconsistency Detection

In the IoT individual nodes may be compromised by an attacker and later used to launch multiple attacks. For example, in RPL-based 6LoWPAN networks the attacker can use compromised nodes to send wrong information about their rank or one of their neighbor's rank to the 6Mapper. It is also possible to get an incorrect or inconsistent view of the network because of the lossy links in the IoT. It is therefore important to detect the inconsistencies, distinguish between valid and invalid consistencies, and correct the invalid information. The complete algorithm to detect and correct the routing graph inconsistencies is described in Algorithm 1.

In order to *detect* incorrect information and to make sure that information is consistent across the network, each edge in the network is checked. The 6Mapper provides node ID and rank of each node, of its parents, and of its neighbors. We iterate over each edge in the network, checking that both nodes agree with each other about their rank and detect the inconsistencies. It is possible that a false alarm is raised because the detected incorrect information is a result of valid mapping inconsistencies described in Section 10.3.1.

In order to *distinguish* between valid and invalid inconsistencies, or to avoid false positives, we rely on *(i)* the number of reported faulty ranks and *(ii)* the difference between the two reported ranks. We use a simple threshold, referred to as $FaultThreshold$ in Algorithm 1, and classify a node as faulty if the number of disagreements this node has with other nodes are larger than the threshold. Most of the disagreements between two nodes are small and a result of varying link quality and ultimate RPL adjustments. To accommodate valid inconsistencies, we only consider disagreements where the difference of the two nodes ranks is greater than 20% of the ranks average; this value is based on our empirical evaluation of SVELTE.

Algorithm 1 Detect and Correct the RPL DODAG Inconsistencies

Require: N - A list of nodes
  for $Node$ in N **do**
    for $Neighbor$ in $Node.neighbors$ **do**
      $Diff = |Node.\text{neighborRank}(Neighbor) - Neighbor.rank|$
      $Avg = (Node.\text{neighborRank}(Neighbor) + Neighbor.rank)/2$ {If the absolute difference is greater than 20% of the ranks average}
      if $Diff > Avg * 0.2$ **then**
        $Node.fault = Node.fault + 1$
        $Neighbor.fault = Neighbor.fault + 1$
      end if
    end for
  end for
  for $Node$ in N **do**
    if $Node.fault > FaultThreshold$ **then**
      $Node.rank =$ Rank reported for $Node$ by any neighbor
      for $Neighbor$ in $Node.neighbors$ **do**
        $Node.\text{neighborRank}(Neighbor) = Neighbor.rank$
      end for
    end if
  end for

We *correct* the faulty information when both of the above conditions are met, i.e., once we have large inconsistencies towards a node. The faulty information corresponding to a node is corrected by changing the rank known to 6Mapper by substituting it with the information reported by one of its neighbors. The neighbor information is updated with the information reported directly by its neighbors.

Once it is detected that a routing inconsistency is a result of a deliberate attack, SVELTE either removes the faulty node or corrects the inconsistency. SVELTE keeps track of inconsistencies and if it is the first time a node is detected as malicious it is not immediately removed as it may be a false alarm or result of a passive attack; in this case the faulty information is corrected as described above. However, if the same node is detected as faulty again it is removed by deleting its entry from the whitelist maintained in the 6Mapper.

Checking Node Availability

It is important to detect if a node or set of nodes are available and operating properly. When a particular node is compromised it may launch multiple attacks to disrupt the network. For example, it may launch a selective forwarding attack and intelligently drop messages. If an RPL network uses CoAP to send

Algorithm 2 Detect Filtered Nodes

Require: W - Set of whitelisted nodes
Require: M - Set of nodes known to the 6Mapper
  $F = []$ {F will contain the filtered nodes}
  for $Node$ in W **do**
    if $Node$ in M **and** $M[Node]$.lastUpdate() $> RecencyThreshold$ **then**
      F.add($Node$)
    end if
  end for
  return F

application data the attacker could forward RPL traffic but drop CoAP traffic. This would result in a seemingly working network even though no useful traffic gets through.

Depending on the RPL implementation and the configuration, we can use the RPL routing table in the RPL DODAG root as a basis for available nodes in the network. As we require a whitelist of valid nodes in the network for access control we could also use that list as a basis for detection.

When we compare the whitelisted nodes with the nodes in our RPL DODAG all differences are offline nodes or unauthorized nodes. Let W be a set of all whitelisted nodes and let R be the nodes known to RPL in the RPL DODAG root, the offline nodes, O are thus:

$$W \setminus R = O$$

where O is the relative complement ($\setminus$) between two sets W and R meaning that O contains all elements of W that are not in R.

It is however not possible to determine if nodes excluded from O are being filtered or are simply offline. That is, if an attacker performs a selective forwarding attack and filters everything but RPL messages it would with the previous method appear as if the nodes are still online, even though all application data is being filtered. By extending the above method with the information available through 6Mapper it is also possible to detect selective forwarding attacks. Let M represent nodes known to 6Mapper and F be the filtered nodes we get the following relationship:

$$W \setminus M = F$$

As the 6Mapper for each node keeps track of the last time it received a packet from a node we can detect filtered nodes by simply checking if we have not recently received any packets from them. In order to mitigate the effects

of packet loss or other similar events common in lossy networks we introduce a threshold on the time since our last packet. We define the threshold as a number of mapping-requests allowed to be unanswered. With this threshold it is possible to alter the sensitivity of the filtered node detection to be easily adaptable to specific deployments. Algorithm 2 describes this behavior and finds all filtered nodes F in a network.

Routing Graph Validity

By artificially altering the routing graph, an attacker can reshape the topology of the network and can control the traffic flow to his advantage. For example, an attacker performs a sinkhole attack by advertising a very good rank to its neighbors. The problem becomes more severe if the sinkhole attack is coupled with other attacks. A sinkhole attack can, for example, enable the attacker to intercept and potentially alter more traffic than otherwise. If combined with a selective forwarding attack a much larger part of the network can be controlled. It is therefore important to detect such attacks.

With SVELTE, it is possible to detect most sinkhole attacks by analyzing the network topology. If the routing graph is inconsistent it is likely an attack is in place. In RPL, the rank in the network should be decreasing towards the root, i.e., in any child-parent relation the parent should always have a lower rank than the child. All cases where a child has a better rank than its parent is an indication of routing graph incoherency, as specified in [7].

When an incoherency is found the child in the relation is at fault, as a node should never have a lower rank than its parent. With such a simple approach false positives are likely to arise, i.e., we detect inconsistencies while in fact all nodes are working properly.

In order to minimize the effects of valid inconsistencies, that can raise false positives, we require several consecutive inconsistencies to be reported for the same nodes. That is we require more than one sample of the network to have the same incoherency to raise an alarm. This is described in the Algorithm 3 as $FaultThreshold$ which is a global state kept between consecutive runs of the detection algorithm. In RPL the rank between any host and its parent is at least $MinHopRankIncrease$ [7]. We utilize this in our algorithm to better conform to the RPL standard.

A sinkhole attack would in most cases be detected by this algorithm. As the attacker advertises a beneficial rank it will most likely have to advertise a better rank than its parent and as such would be detected by the detection scheme described above. If a sinkhole attack is to remain undetected the advertised

Algorithm 3 Finding Rank Inconsistencies

Require: N - A list of nodes
  for $Node$ in N **do**
    if $Node.rank + MinHopRankIncrease < Node.parent.rank$ **then**
      $Node.fault = Node.fault + 1$
    end if
  end for
  for $Node$ in N **do**
    if $Node.fault > FaultThreshold$ **then**
      Raise alarm
    end if
  end for

Algorithm 4 Adapt to End-to-end Losses

Require: $dest$ - The destination with packet loss
  $nexthop$ =getNexthop($dest$)
  $nexthop.metric = nexthop.metric * 0.8$

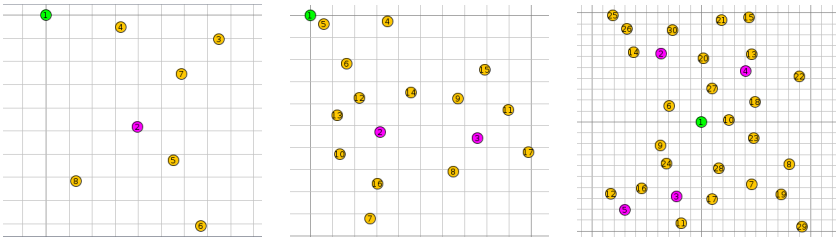
rank of a malicious node must not be better than that of its parent. This would in turn result in the adversary's rank only being slightly improved over a non-adversarial node and thus yield little benefit.

In RPL, the rank as well as the parent selection is calculated via an objective function, which might use factors such as link quality in its calculation; for example when the Expected Transmission Count (ETX) [13] is used to calculate rank. The ETX is an approximation of the link quality and as such a bad link might affect the choice of parent more than a slight difference in rank. This would further lower the impact of a sinkhole attack that is undetectable by Algorithm 3.

End-to-end Packet Loss Adaptation

We design an intentionally simple system to take end to end losses into account when calculating the route and to mitigate the effects of filtering hosts. If a reliable higher layer protocol such as TCP or CoAP (with *confirmable* messaging) is used, packet loss can be detected using the protocol's acknowledgement mechanism. The reasoning behind a host-to-host packet loss indication is that if an attacker is filtering packets some hops down the path we want to be able to adapt to it. In the RPL-based network, if a packet is filtered somewhere on the path a new parent should eventually be tried.

The approach is not able to adapt to every form of filtering, for example,



(a) 8 nodes with 1 (ID 1) malicious node.

(b) 16 nodes with 2 (ID 2 and 3) malicious nodes.

(c) 32 nodes with 4 (ID 2, 3, 4, 5) malicious nodes.

Figure 10.4: Network configurations and node placement that are used in the experiments in this section

when the attacker is located such that all packets have to go through it. If however a collection-scheme with acknowledgements is also running in the network all data losses should be corrected for. Since all nodes will try to send data to the sink all nodes with a path through the attacker will also notice the losses and correct for them, given that the attacker filters all application data. If a packet is not able to reach its destination, we slightly alter the route metric of the route, that is the next-hop neighbor for that packet, (20% in Algorithm 4) to reflect that there might be an attacker along the path. Algorithm 4 describes end-to-end packet loss adaptation.

Sybil and CloneID Attacks Protection

In a Sybil attack an attacker copies several logical identities on one physical node whereas in a cloned identity (CloneID) attack the attacker copies the same logical identity on several physical nodes. Both attacks are aimed to gain access to a large part of the network or in order to overcome a voting scheme. The 6Mapper only considers the latest information received from each host in the network where a host is identified by an IP address. A sybil attack has no direct effect on the 6Mapper as it makes no difference if the identities are on the same physical node as if they are separate physical entities, each host is treated individually in both cases. While cloned identities can interrupt the routing in a network it does not affect the 6Mapper directly as the 6Mapper only considers the latest information received from one of the identity. As a result if two cloned nodes send information to the 6Mapper there is no difference compared to if one node sends the information twice, thus not directly affecting

Algorithm 5 Mini-firewall

Require: $Host$ - The host to report
Require: $Source$ - The node that sent the report
Require: $GlobalFilter$ - A set of external hosts to filter towards all nodes
Require: $LocalFilter$ - A map mapping an external host to a set of local nodes. The set describes all nodes that have reported that specific external host.

```
if Host in GlobalFilter then
    return Host already filtered
end if
if Host in LocalFilter then
    Filter = LocalFilter.get(Host)
    {Add Source to the list of nodes blaming Host}
    Filter.add(Source)
    if Filter.size() ≥ ReportThreshold then
        GlobalFilter.add(Host)
        LocalFilter.remove(Host)
    end if
end if
```

the operations of the 6Mapper. Sybil attacks and cloned identities are both often used to disrupt different voting schemes by giving an attacker more votes. Voting schemes based upon 6Mapper collected data will be unhindered by both sybil attacks and cloned identities.

10.3.3 Distributed Mini-firewall

Though SVELTE can protect 6LoWPAN networks against in-network intrusion, it is also important that the resource constrained nodes are protected against global attackers that are much more powerful. For example, it is easier for hosts on the Internet than constrained nodes in 6LoWPAN networks to perform denial of service attacks. Firewalls are usually used to filter external hosts and/or messages destined to local networks. As the end-to-end message confidentiality and integrity is necessary in the IoT, the SVELTE module in the 6BR or a firewall cannot inspect the contents of the encrypted messages; therefore, it is hard to distinguish between the legitimate and malicious external traffic.

We propose a distributed mini-firewall that protects a 6LoWPAN network from external hosts. The firewall has a module in the 6BR and in the constrained nodes, and is integrated with SVELTE. Our firewall, besides providing typical blocking functionally against well-known external attackers specified manually by the network administrator, can block the external malicious hosts specified in *real-time* by the nodes inside a 6LoWPAN network.

The destination host inside a 6LoWPAN node can see the encrypted contents and hence analyze the malicious traffic and notify the 6BR in real-time to filter traffic coming from the compromised host, therefore stopping the traffic before it reaches the constrained nodes. When a constrained node notices an external host being abusive it sends a packet with the host IP to the firewall module in the 6BR. As is the case with the 6Mapper, if IPsec with Authentication Header is used the nodes own ID can be omitted. Otherwise, the nodes own ID need to be included. If the node ID is included it can be compressed down to 2 bytes using 6LoWPAN header compression mechanisms. The external host however can neither be compressed nor omitted as it can be any valid IPv6 address. Therefore the minimal size of the filtering-request packet is 16 bytes. With the node ID the size of the packet is 18 bytes.

In order to make sure that no internal compromised node can abuse this mechanism by requesting filtering of traffic from a legitimate external host, both the source and the destination is taken into account when filtering. The node inside a 6LoWPAN network can only choose to filter the traffic destined to itself. Such a firewall is still easy to circumvent as the attacker can simply target another node in the network and start the attack again; therefore, we extend the firewall to adapt and block any external host if a minimum set of nodes complain about the same external host. Our mini-firewall is described in Algorithm 5.

To be more *preventive* against global attackers, our mini-firewall can be extended with AEGIS [14], a rule-based firewall for wireless sensor networks.

10.4 Implementation

We implement SVELTE and the mini-firewall in the Contiki OS [9], a well known operating system for the IoT. Contiki has a well tested implementation of RPL (ContikiRPL). As SVELTE is primarily designed to detect routing attacks we make use of the RPL implementation in the Contiki operating system to develop the 6Mapper, the firewall, and the intrusion detection modules. The RPL implementation in Contiki utilizes in-network routing where each node keeps track of all its descendants. We borrow this feature to detect which nodes should be available in the network. To provide IP communication in 6LoWPAN we use μIP, an IP stack in Contiki, and *SICSLoWPAN*- the Contiki implementation of 6LoWPAN header compression. We also implement the sinkhole and selective forwarding attacks against RPL to evaluate SVELTE.

SVELTE is open source[2] and is available to researchers and industry.

10.5 Evaluation

In this section we present the empirical evaluation of SVELTE. After describing our experimental setup, we quantitatively evaluate the detection rate and the true positives for each experiment. We also measure the overhead of SVELTE both at the node-level and network-wide. We evaluate the overhead in terms of energy consumption and the memory footprint.

10.5.1 Experimental Setup

We run our experiments in Contiki's network simulator Cooja [15] that has shown to produce realistic results [16]. Cooja runs deployable Contiki code. In our simulations, we use emulated Tmote Sky [17] nodes.

In general, we expect that the 6BR is not a constrained node and it can be a PC or a laptop; however, currently there exists no PC equivalent 802.15.4 devices, therefore we run the 6Mapper natively, i.e., on Linux, and communicate with Cooja using a *serial socket*. For RPL with 6Mapper we run each test 10 times, and calculate the average and standard deviation to show the accuracy and precision of our results. On the other hand, the experiments with RPL only (without the 6Mapper) have no processing intensive components and hence require no native parts. Therefore, the experiments with RPL-only yield the same results for all experiments as we use the same seed.

10.5.2 SVELTE Detection and True Positive Rate

Here we quantitatively evaluate the detection rate, i.e., the number of malicious nodes successfully detected against the total number of malicious nodes present in the system, and the true positives rate, i.e., the total number of successful alarms divided by the total number of alarms. We use three different configurations shown in Figure 10.4a, 10.4b and 10.4c. In each configuration node no 1 (green) is the 6BR. Using these settings, we run experiments for 5, 10, 20, and 30 minutes. In all experiments, the 6Mapper is configured to request data and to perform analysis every two minutes. Therefore, the first 6Mapper request will be sent after two minutes. The first analysis is also performed

[2]For the source code visit: `http://www.shahidraza.info`

after 2 minutes but will however not yield any results as no data is yet gathered. Therefore the earliest possible detection time is after four minutes. It is important to note that these are the settings in our experiments and not the requirements for SVELTE. The malicious nodes can spoof or alter information, and/or can perform sinkhole or selective forwarding attacks. In the following experiments SVELTE first performs network graph inconsistency detection as described in Section 10.3.2, before detecting sinkhole or selective forwarding attacks. Each experiment is run in a lossy and in a lossless network. Lossless links provide the perfect scenario for 6Mapper, as all requests and responses return without delay and loss, and we get a true picture of the network. This is further improved by the fact that nodes more quickly can propagate their ranks down in the network graph. The real 6LoWPAN networks are mostly lossy, therefore we consider both cases in our evaluation. The loss model is Cooja's default radio model that uses a Unit Disk Graph Medium (UDGM): Distance Loss [15]. UDGM models the transmission range as a circle in which only the nodes inside the circle receive packets. The UDGM Distance Loss model, an extension of UDGM, also considers interference.

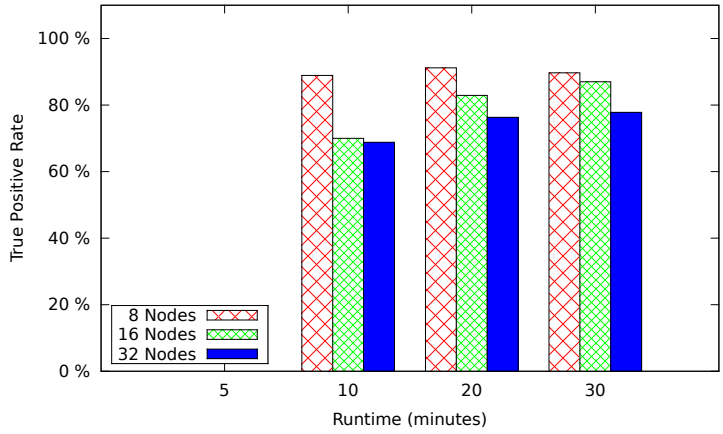


Figure 10.5: For the smaller lossy network, SVELTE has 90% true positive rate against sinkhole attacks which decreases for larger networks but gets better when RPL becomes stable.

Sinkhole Attacks with and without losses The results for the sinkhole attack in a lossless network scenario show almost 100% true positive rate on the first possible attempt to analyze the network and no false positives are detected during the simulations. A lossless network configuration means that all requested data is gathered quickly and without losses, which implies that the map of the network is a perfect representation of the actual network. Because of this it is very easy to detect all sinkhole attacks without any false positives. In the lossy network configuration, Figure 10.4a, the true alarm rate is approximately 90%, as shown in Figure 10.5. However, with the increase in network size the true alarm rate decreases; this is because for the larger network configurations it takes some time before the RPL network and our map of the network become stable and complete enough to arrive at a higher true positive rate. For example, in the scenario with 16 nodes it takes 30 minutes to arrive at the same true positive rate as is done with 8 nodes after 10 minutes. The reason Figure 10.5 shows a non-existent detection rate for the case of 5 minutes is because we only raise an alarm if the same node has been misbehaving for more than two consecutive executions of our algorithm. Hence, the current configuration implies that a sinkhole attack can be detected after 6 minutes. Our approach does not require collection of two consecutive messages or executions to work. Collecting multiple messages is advantageous to make sure that it was actually an attack and not a sudden link fluctuations, for example due to interference. If the attack persists for two consecutive executions of our algorithms then we raise an alarm; this is done primarily to reduce false positives.

From these results it is evident that SVELTE is very effective against sinkhole attacks in a network with no or few losses, and in lossy networks it is more effective when the RPL network has become stable.

Selective Forwarding Attack with and without losses In a selective forwarding attack a malicious node filters traffic going through it. Hence, the 6Mapper will not be able to get any data from any children of the malicious nodes in the network. This in turn has the effect that the results of the 6Mapper depend on the actual network topology, i.e., in the lossless case, unlike with sinkhole attacks, the results are not always 100%. We can see the effects of this phenomenon in Figure 10.6a. In a lossy network, as shown in Figure 10.6b, there is a gradual increase in the true positive rate going towards a bit over 80% in all cases. As the network is lossy messages are naturally lost, and if that happens several consecutive times when mapping we are going to get more false positives. If we raise the various thresholds in our detection algorithms it is possible to lower the number of false alarms, possibly at the cost

Typical Conditions	MIN	NOM	MAX	UNIT
Voltage	2.1		3.6	V
Free air Temperature	-40		85	C
MCU on, Radio RX		21.8	23	mA
MCU on, Radio TX		19.5	21	mA
MCU on, Radio off		1800	2400	μA
MCU idle, Radio off		54.5	1200	μA
MCU standby		5.1	21.0	μA

Table 10.1: Operating Conditions in Tmote sky

of a decreased detection rate. In order to reduce number of false positives we may use location information of the nodes as discussed in Section 10.7.

We also measure the detection rate during all of the above experiments. We achieve 100% *detection rate* meaning that we can detect all malicious nodes that launch sinkhole and/or selective forwarding attacks. It should be noted that the 100% detection rate is for the current set of experiments with the current setting; we do not claim that SVELTE should achieve 100% detection rate in all settings. As can be seen in Figure 10.5 and 10.6 the *true positive rate* is not 100%, i.e., we have some false alarms during the detection of malicious nodes. This is mostly caused by our configuration. It might be possible to alter the behavior of our detection algorithm, for example, by changing the threshold used in Algorithm 2 and thus possibly get a different result with regards to detection rate and/or false alarm rate.

10.5.3 Energy Overhead

The nodes in the IoT are usually battery powered and hence energy is a scarce resource. Here we measure SVELTE's power consumption both at node-level and at system-level. We use Contiki Powertrace [18] to measure the power consumption. The output from the Powertrace application is the total time the different parts of the system were on.

We calculate the energy usage and power consumption using the nominal values, the typical operating conditions of the Tmote sky, shown in Table 10.1. We use 3V in our calculations. In the rest of this paper MCU idle while the radio is off is referred to as low power mode, or LPM. The time the MCU is on and the radio is off is referred to as CPU time. The time the radio is receiving and transmitting with the MCU on is referred to as $listen$ and $transmit$ respectively. We measure energy in both duty cycled 6LoWPAN

networks, where the radio is mostly off, and in non duty cycled networks where the radio is always on for listening and transmitting.

Network-wide with Duty Cycling

Here we evaluate network-wide energy consumption of an RPL network with and without the 6Mapper and intrusion detection mechanisms in a duty cycled network. We use ContikiMAC [19], a duty cycling MAC protocol in Contiki. We use the default ContikiMAC setting that has 8 wakeups per second and without traffic the radio is on for 0.6% of the time. We run each experiment in a network of 8, 16, 32 and 64 emulated Tmote sky nodes, with nodes placed at the same locations.

Figure 10.7a shows the network-wide energy usage for 30 minutes by all the nodes, calculated as follows

$$Energy(mJ) = (transmit * 19.5mA + listen * 21.8mA + CPU* \\ 1.8mA + LPM * 0.0545mA) * 3V/4096 * 8$$

From the network wide energy usage, we calculate the average power as,

$$Power(mW) = \frac{Energy(mJ)}{Time(s)}$$

which when divided by the total number of nodes gives us the per node average power consumption during the experiment. Figure 10.7b shows the power consumption per node. As can be seen in Figure 10.7a and 10.7b the overhead of the 6Mapper is negligible for small networks (up to 16 nodes) and increases with the number of nodes. The total overhead of SVELTE is approximately 30% more than running RPL only for networks with 64 nodes. Recall that with duty cycling the radio is off for approximately 99% of the time.

Network-wide without Duty Cycling

We use the same network settings as in Section 10.5.3 and run the experiments in a non duty cycled network where the radio is always turned on to receive and transmit packets. When we compare the results of RPL with the 6Mapper plus intrusion detection algorithms we see that the overhead is negligible. This is because the radio is always on and most of the nodes' energy is consumed on idle listening.

In-node Energy Overhead

Here we measure the energy consumption of handling a single event of the 6Mapper and the firewall inside a constrained node. Table 10.2 lists the energy

Event	Energy (mJ)
6Mapper Response Handling	0.1465
Firewall handling	0.0478
Packet lost correction	0.0483

Table 10.2: Energy consumption for handling a single event inside a constrained node.

Configuration	Total ROM ($byte$)	Overhead ($byte$)
6Mapper client	44 264	1 414
Firewall client	43 556	0 246
Packet loss improvement	43 264	0 122
6Mapper server (1 node, 1 neighbor)	46 798	3 580
6Mapper server (8 node, 1 neighbor)	46 798	3 846
6Mapper server (16 nodes, 1 neighbor)	46 800	4 152
6Mapper server (16 nodes, 8 neighbors)	46 924	4 724

Table 10.3: Out of total $48k$ of ROM size in a constrained device (Tmoke sky), SVELTE requires $1.76k$. However, in the 6BR (typically a PC) the size grows when the number of nodes increases.

required to perform different tasks; this does not include the energy needed to send/receive packets which we have included in Section 10.5.3 and 10.5.3. As can be seen in Table 10.2, a constrained node consumes very little energy for local processing as most of the processing intensive tasks are performed in the 6BR where the 6Mapper and the main SVELTE detection modules reside. Therefore, the energy consumed for in-node processing is clearly negligible.

10.5.4 Memory Consumption

In Table 10.3 we show the extra ROM requirements of SVELTE's different modules. The baseline for each configuration is different as some depend on different parts of the Contiki system. For example, the 6Mapper that resides in the 6BR (typically a PC) requires more ROM than other nodes. However, the total additional ROM required to host SVELTE's modules inside a constrained node is $1.76k$ which is well below the total available ROM in constrained devices such as $48k$ in Tmote sky. In Table 10.3 it is important to note the overhead column which shows the pure overhead of SVELTE modules in Contiki. Even though 6Mapper is not targeted towards running on constrained nodes it

Event	RAM ($byte$)
6Mapper Response Handling	162
Firewall handling	24
Packet lost correction	188

Table 10.4: Additional RAM usage by SVELTE for handling a single event inside a constrained node.

is still lightweight enough and can be used for small networks.

We also measure the RAM size of 6Mapper response handling, firewall, and packet loss correction which we show in Table 10.4. The total RAM size in the Tmoke sky is $10kb$, hence SVELTE modules with $0.365k$ additional RAM requirement can easily run in constrained nodes.

10.6 Related Work

The IoT is a rather old concept and for many years RFID-based sensors were considered as *things* in the IoT. With the inception of 6LoWPAN, lightweight IP is being standardized and used in the IoT for the unique identification and global connectivity of the *things*. Even when confidentiality and integrity are enforced by message security solutions such as IPsec [4] it is possible to disrupt the IoT. A number of attacks against the IoT have been identified [8] in addition to those against WSN [12] that are also applicable to the IoT. Therefore, it is important to have systems that detect such attacks.

The concept of intrusion detection is quite old and extensive research is carried out in this field mostly against the Internet attacks and attacks against WSN. However, no IDS are specifically designed in the context of IoT. Most of the IDS approaches for WSN are based on a distributed architecture and are built on the limitation that there is no centralized management and control point. A common IDS approach for WSNs is to utilize several special nodes distributed evenly throughout the network. These special nodes can either be physically different [20] or dynamically distributed throughout the network [21, 22]. In real deployments, however, it cannot be guaranteed that particular nodes are always present in specific locations in the network; also, the cost of employing mobile agents that move through the network might be too high. Clustering based approaches have similar issues as each cluster often requires a powerful entity for coordination [23]. The IoT has a novel architecture where the 6BR is always assumed to be accessible and is a potential place

for centralized management and control. SVELTE make use of this novel IoT architecture and presents a new placement for IDS. Using a mix of centralized and distributed architecture SVELTE takes advantage of both realms.

Many IDS approaches are based upon watchdog techniques [21, 24] which could be used in the IoT. In addition to being distributed and fully deployed on sensor nodes, a general problem with watchdog based approaches is that they require promiscuous listening, which consumes a lot of power and therefore is not suitable for constrained devices. Advanced anomaly detection approaches are proposed [25, 26], not primarily for WSNs, which on one hand can detect many intrusions efficiently but on the other hand requires intelligent learning, which is both expensive and difficult in low powered 6LoWPAN networks.

Most current IDS approaches require different routing schemes that are not based on standardized mechanisms. As far as we are aware, no approach is built around 6LoWPAN and RPL in the context of the IoT. Our approach considers RPL to decrease the cost of performing intrusion detection. Likewise, we have taken into account the fact that there is a central entity, the 6BR, that connects the sensor network with the conventional Internet, which is a standard based networking solution [1, 2, 7].

We do not claim that no other IDS approach can be used in the RPL-connected IoT. Rather we argue that these approaches are built on different assumptions that do not fully hold in the IoT architecture. Also, the IoT gives rise to new challenges that do not exist in typical WSNs. However, there is a potential to incorporate already available approaches in the SVELTE *architecture*. We discuss below the possibilities to integrate available lightweight IDS approaches in SVELTE.

10.7 SVELTE Extensions

One of the main advantages of our approach to intrusion detection is that the proposed and developed system is very easy to extend. There are a number of potential attacks against the Internet of Things and it is likely that more attacks will be discovered. As such extendability is very important for an IDS. The 6Mapper is easy to extend both conceptually and in practice. If a new detection scheme requires more data to be added to the network graph the response packets can easily be extended. Also, using the already available data that we collect through the 6Mapper it is possible to apply anomaly detection techniques, for example via the use of Support Vector Machines [27], feature vectors [28], or automata based approach [29].

Wormhole Detection One of the important to detect attacks in wireless networks is wormhole [30]. If the 6Mapper is extended with the signal strength of each node's neighbor it is also possible to detect wormhole attacks [31].
Pinpointing filtering node If a node is filtering traffic it is beneficial to be able to pinpoint more accurately which node is performing the filtering. The most straight forward approach is to perform a traceroute [32] towards one of the missing nodes.
Location Information RPL is primarily designed for static networks, though it can be extended to support mobility [33], it is possible to add node's location in the 6Mapper at the deployment time. The location of a node can also be estimated in real-time using localization techniques [34]. These location information help SVELTE to build a physical map of the network that will ultimately enhance its intrusion detection capabilities. For instance, with this physical map rank modification and hence the sink-hole attack can be detected with even lesser false positives alarms. The location information of nodes will also help SVELTE to mitigate the sybil and CloneID attacks aimed to disrupt the routing information [35].

10.8 Conclusions

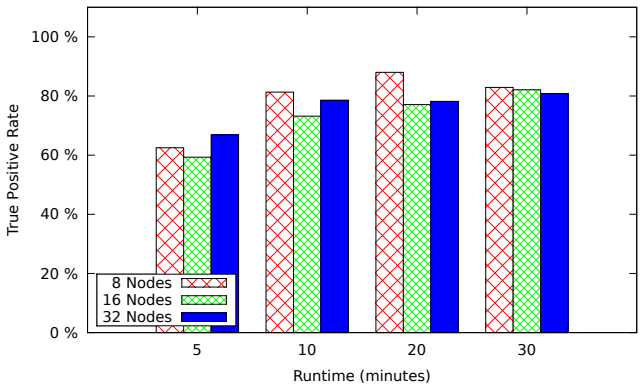
6LoWPAN networks will be an integral part of the IoT. Considering the potential applications of the IoT it is important that 6LoWPAN networks are protected against internal and external intrusions. To this end we present SVELTE, the first IDS for the IoT which consists of a novel architecture and intrusion detection algorithms. We implement and evaluate SVELTE and show that it is indeed feasible to use it in the context of RPL, 6LoWPAN, and the IoT. To guard against global attacks we also design and implement a mini-firewall.

The detection algorithms in SVELTE currently target spoofed or altered information, sinkhole and selective forwarding attacks. However, it is flexible and can be extended to detect more attacks. Therefore, we plan to complement SVELTE with novel and/or available intrusion detection techniques that are feasible to use in the context of the IoT.

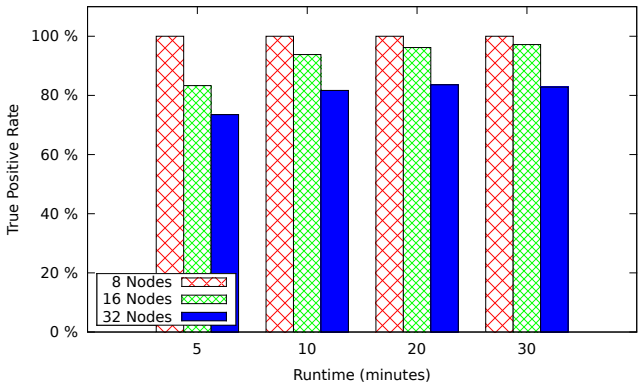
Acknowledgements

This work was financed by the SICS Center for Networked Systems (CNS), SSF through the Promos project, and CALIPSO, Connect All IP-based Smart

Objects, funded by the European Commission under FP7 with contract number FP7-ICT-2011.1.3-288879.

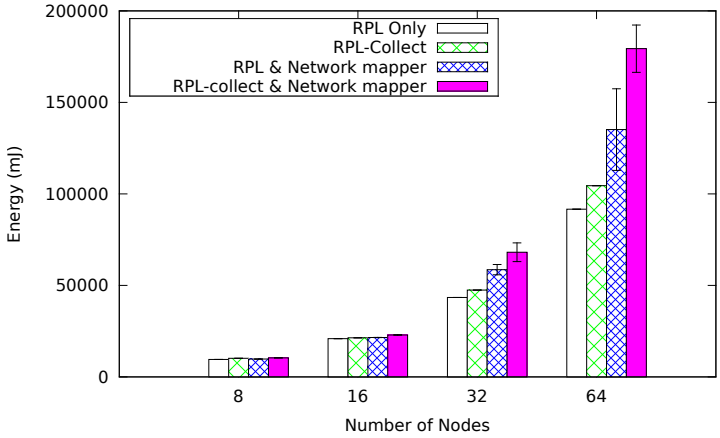


(a) *Lossy* network suffering from selective forwarding attack

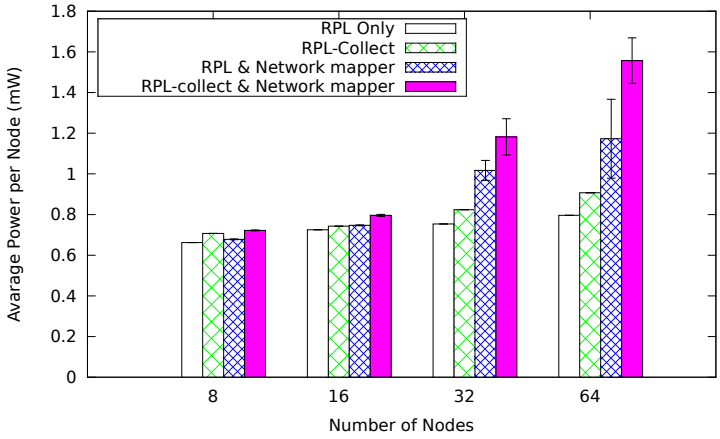


(b) *Lossless* network suffering from selective forwarding attack

Figure 10.6: SVELTE has acceptable true positive rate in both lossy and lossless network considering that we have almost 100% detection rate for selective forwarding attacks.



(a) Energy usage for the entire network (with *duty cycling*) in 30 minutes.



(b) Average power consumption per node in a *duty cycled* RPL-based network

Figure 10.7: Network-wide energy usage in a *duty cycled* RPL-based network of different sizes shows that for network with less nodes SVELTE overhead is very small; however, per node overhead grows with the increase in number of nodes as more and more nodes act as routers.

Bibliography

[1] J. Hui and P. Thubert. Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks. RFC 6282, September 2011.

[2] 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.

[3] T. Kothmayr, W. Hu, C. Schmitt, M. Bruenig, and G. Carle. Securing the internet of things with dtls. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, pages 345–346. ACM, 2011.

[4] Shahid Raza, Simon Duquennoy, A. Chung, Dogan Yazar, Thimo Voigt, and Utz Roedig. Securing communication in 6lowpan with compressed ipsec. In *7th International Conference on Distributed Computing in Sensor Systems (DCOSS'11)*, pages 1–8, Barcelona, Spain, 2011.

[5] Shahid Raza, Simon Duquennoy, Joel Höglund, Utz Roedig, and Thiemo Voigt. Secure Communication for the Internet of Things - A Comparison of Link-Layer Security and IPsec for 6LoWPAN. *Security and Communication Networks, Wiley.*, January 2012.

[6] Z. Shelby, K. Kartke, C. Bormann, and B. Frank. Constrained Application Protocol (CoAP). draft-ietf-core-coap-12, October 2012.

[7] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander. RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. RFC 6550, March 2012.

[8] O. Garcia-Morchon, R. Hummen, S.S. Kumar, R. Struik, and S.L. Keoh. Security Considerations in the IP-based Internet of Things. draft-garcia-core-security-04, March 2012.

[9] A. Dunkels, B. Grönvall, and T. Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *EMNets'04*, pages 455–462, Tampa, USA, November 2004.

[10] S. Kent and R. Atkinson. IP Encapsulating Security Payload (ESP). RFC 2406, November 1998. Obsoleted by RFCs 4303, 4305.

[11] S. Kent and R. Atkinson. IP Authentication Header. RFC 2402, November 1998. Obsoleted by RFCs 4302, 4305.

[12] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. *Ad hoc networks*, 1(2):293–315, 2003.

[13] D.S.J.D. Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. *Wireless Networks*, 11(4):419–434, 2005.

[14] M. Hossain and V. Raghunathan. Aegis: A lightweight firewall for wireless sensor networks. *Distributed Computing in Sensor Systems*, pages 258–272, 2010.

[15] Fredrik Österlind, Adam Dunkels, Joakim Eriksson, Niclas Finne, and Thiemo Voigt. Cross-level sensor network simulation with cooja. In *Proceedings of 31st IEEE Conference on Local Computer Networks*, pages 641–648. IEEE, 2006.

[16] F. Österlind. *Improving Low-Power Wireless Protocols with Timing-Accurate Simulation*. PhD thesis, Uppsala University, 2011.

[17] J. Polastre, R. Szewczyk, and D. Culler. Telos: enabling ultra-low power wireless research. In *IPSN'05*, pages 364–369, April 2005.

[18] A. Dunkels, J. Eriksson, N. Finne, and N. Tsiftes. Powertrace: Network-level power profiling for low-power wireless networks, 2011.

[19] Adam Dunkels. The contikimac radio duty cycling protocol, 2011.

[20] I.M. Atakli, H. Hu, Y. Chen, W.S. Ku, and Z. Su. Malicious node detection in wireless sensor networks using weighted trust evaluation. In *Proceedings of the 2008 Spring simulation multiconference*, pages 836–843. Society for Computer Simulation International, 2008.

[21] R. Roman, J. Zhou, and J. Lopez. Applying intrusion detection systems to wireless sensor networks. In *Proceedings of IEEE Consumer Communications and Networking Conference*, pages 640–644, 2006.

[22] T.H. Hai, E.N. Huh, and M. Jo. A lightweight intrusion detection framework for wireless sensor networks. *Wireless Communications and mobile computing*, 10(4):559–572, 2009.

[23] C. Rong, S. Eggen, and H. Cheng. An efficient intrusion detection scheme for wireless sensor networks. *Secure and Trust Computing, Data Management, and Applications*, 187:116–129, 2011.

[24] Sergio Marti, T. J. Giuli, Kevin Lai, and Mary Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, MobiCom '00, pages 255–265, New York, NY, USA, 2000. ACM.

[25] Amitabh Mishra, Ketan Nadkarni, and Animesh Patcha. Intrusion detection in wireless ad hoc networks. *Wireless Communications, IEEE*, 11(1):48–60, 2004.

[26] Kai Hwang, Min Cai, Ying Chen, and Min Qin. Hybrid intrusion detection with weighted signature generation over anomalous internet episodes. *Dependable and Secure Computing, IEEE Transactions on*, 4(1):41–55, 2007.

[27] S. Kaplantzis, A. Shilton, N. Mani, and Y.A. Sekercioglu. Detecting selective forwarding attacks in wireless sensor networks using support vector machines. In *Intelligent Sensors, Sensor Networks and Information, 2007. ISSNIP 2007. 3rd International Conference on*, pages 335–340. IEEE, 2007.

[28] M.A. Livani and M. Abadi. A pca-based distributed approach for intrusion detection in wireless sensor networks. In *International Symposium on Computer Networks and Distributed Systems (CNDS)*, pages 55–60. IEEE, 2011.

[29] S. Misra, K.I. Abraham, M.S. Obaidat, and P.V. Krishna. Laid: a learning automata-based scheme for intrusion detection in wireless sensor networks. *Security and Communication Networks*, 2(2):105–115, 2008.

[30] Y.C. Hu, A. Perrig, and D.B. Johnson. Wormhole attacks in wireless networks. *Selected Areas in Communications, IEEE Journal on*, 24(2):370–380, 2006.

[31] W. Wang and B. Bhargava. Visualization of wormholes in sensor networks. In *Proceedings of the 3rd ACM workshop on Wireless security*, pages 51–60. ACM, 2004.

[32] G. Malkin. Traceroute Using an IP Option. RFC 1393 (Experimental), January 1993.

[33] Kevin C Lee, Raghuram Sudhaakar, Lillian Dai, Sateesh Addepalli, and Mario Gerla. Rpl under mobility. In *Consumer Communications and Networking Conference (CCNC), 2012 IEEE*, pages 300–304. IEEE, 2012.

[34] Yin Chen, Dimitrios Lymberopoulos, Jie Liu, and Bodhi Priyantha. Fm-based indoor localization. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pages 169–182. ACM, 2012.

[35] J. Newsome, E. Shi, D. Song, and A. Perrig. The sybil attack in sensor networks: analysis & defenses. In *Proceedings of the 3rd international symposium on Information processing in sensor networks*, pages 259–268. ACM, 2004.

Chapter 11

Paper F: Combined Secure Storage and Communication for the Internet of Things

Ibrahim Ethem Bagci, Shahid Raza, Tony Chung, Utz Roedig, Thiemo Voigt.
10th IEEE International Conference on Sensing, Communication, and Networking (SECON13), June 24-27, 2013, New Orleans, USA.

Abstract

The future Internet of Things (IoT) may be based on the existing and established Internet Protocol (IP). Many IoT application scenarios will handle sensitive data. However, as security requirements for storage and communication are addressed separately, work such as key management or cryptographic processing is duplicated. In this paper we present a framework that allows us to combine secure storage and secure communication in the IP-based IoT. We show how data can be stored securely such that it can be delivered securely upon request without further cryptographic processing. Our prototype implementation shows that combined secure storage and communication can reduce the security-related processing on nodes by up to 71% and energy consumption by up to 32.1%.

11.1 Introduction

The Internet of Things (IoT) is becoming a reality and vast numbers of smart objects are interconnected via the Internet Protocol (IP). A number of applications in this context handle sensitive information. For example, smart objects may be used for patient monitoring in hospitals, implementations of security systems in airports or to monitor crucial business processes in factories. Thus, security mechanisms are required to ensure confidentiality, integrity and authenticity of the collected information.

Due to resource limitations of smart objects it is not feasible to use the existing IP protocol throughout the entire IoT. IP header compression, as defined in the 6LoWPAN [1] framework, is used in wireless IEEE 802.15.4 networks which smart objects generally use for interconnectivity. 6LoWPAN header compression and decompression is carried out by gateway nodes when relaying packets between IEEE 802.15.4 networks and the existing IP network infrastructure.

As the IoT relies on the established and tested IP protocol it is reasonable to also use security mechanisms defined in this context. The IPsec [2] framework defines security mechanisms for IP networks and it is supported by nearly all hosts currently in use. A definition of IPsec 6LoWPAN extensions [3] exists which allows smart objects to participate in IPsec secured communication. Thus, secure communication in the IoT using standardised mechanisms is feasible.

Smart objects now provide vast amounts of storage space due to the recent advances in flash memory technology. IoT applications rely on this storage space in order to improve system performance [4]. It is therefore becoming more important to not only secure communication but to also protect sensitive data while it is stored on smart objects. Various secure storage solutions exist that can be used to protect data on nodes. For example, [5] is an extension of the Contiki [6] CFS [7] filesystem that provides security services.

The previously outlined secure communication and storage solutions have been developed individually. It is not taken into account that tasks such as key exchange or cryptographic processing are executed for both system components. Thus, in many situations cryptographic work performed by smart objects is unnecessarily carried out twice or more. Given that smart objects are very resource limited devices it is desirable to prevent such process duplication. Freed resources may be used to reduce hardware complexity, improve energy consumption or to add additional application features.

We address the previously outlined shortcoming of existing solutions and

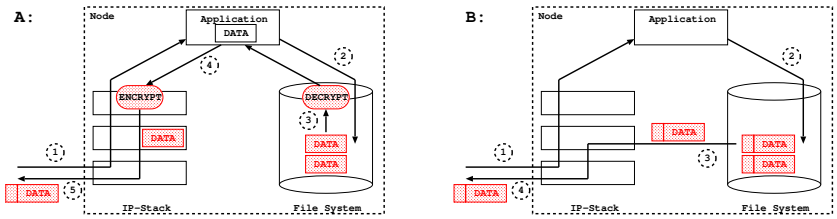


Figure 11.1: *A: Traditional Operation:* 1 - Data is requested from the node. 2 - The application forwards the request to the filesystem. 3 - The data is decrypted and passed to the application. 4 - The application sends data for transmission to the IP stack which secures the data. 5 - The data is transmitted.
B: Combined Secure Storage and Communication: 1 - Data is requested from the node. 2 - The application forwards the request to the filesystem. 3 - The secured data is directly passed into the IP stack. 4 - Data is transmitted without cryptographic processing.

provide a design of a combined secure storage and communication framework that allows us to reduce security related processing on smart objects (see Fig 11.1). In particular we consider the IP, 6LoWPAN and IPsec standards as the base for our work. We believe that a standard compliant solution is more desirable than a proprietary system. Furthermore, it is safer to build on tested and trusted security mechanisms rather than designing an entirely novel mechanisms. Data is stored securely on the flash file system such that it can be directly used for secure transmission. This is not a trivial task as packet header content of future transmissions must be considered when securing data for storage. We show in this paper that an IP based combined secure storage and communication solution is possible and that this can save up to 71% of a node's security related processing effort. A cost in regards to additional storage space is incurred as a result of the secure storage; however, given that smart objects can now provide ample amounts of storage space we do not see this as limiting factor. The specific contributions of this paper are:

- The definition of a framework for combined secure storage and communication for IP/6LoWPAN networks.
- An implementation of the framework for the Contiki operating system.
- A detailed evaluation of the performance gains of the framework.

The next section of the paper discusses related work in the area of secure communication and storage for smart objects. Section 11.3 describes the proposed combined secure storage and communication framework and its implementation for the Contiki OS, which is then evaluated in detail in Section 11.4. Section 11.5 discusses our findings and concludes the paper.

11.2 Related Work

Security in the IoT is a research topic that has attracted a lot of interest. Work has been carried out to improve efficiency of cryptographic algorithms [8], to provide specialised hardware support [9], to organize key distribution [10], to define secure communication protocols [3] and to organise secure data storage [11]. Solutions for secure communication and secure storage of data in the IP based IoT exist, but these functions are generally designed and operated independent of each other. To the best of our knowledge, this is the first work which aims to combine both aspects. Thus, the following shall discuss these IoT security aspects separately.

Secure Communication for the IoT: Communication in the IoT can be secured on different layers. The IoT uses the IEEE 802.15.4 [12] link-layer. IEEE 802.15.4 link-layer security is the current state-of-the-art security solution for the IP-connected IoT; it defines data encryption and integrity verification.

IEEE 802.15.4 security does not provide end-to-end security when connecting a IEEE 802.15.4 network via a gateway router to the existing Internet. Thus, additional solutions exist which protect data traveling from Internet hosts to the border router. For example, ArchRock PhyNET [13] applies IPsec in tunnel mode between the gateway router and Internet hosts.

To achieve true end-to-end security between Internet hosts and smart objects an IPsec extension for 6LoWPAN has been proposed [3]. Unmodified Internet hosts can communicate directly with smart objects. The border router applies 6LoWPAN header compression in order to enable efficient transport of IPsec packets in IEEE 802.15.4 networks. We use this mechanism for our framework.

End-to-end security can be provided by using Transport Layer Security (TLS) or its predecessor Secure Sockets Layer (SSL). SSL has been proposed as security mechanism for the IoT by Hong et al. [14]. Foulagar et al. propose a TLS implementation for smart objects [15].

Secure Storage in the IoT There are a number of secure storage solutions available [5], [11], [16] and [17]. codo is a security extension for the Coffee [7] filesystem in the Contiki [6] OS. ptimises performance of security operations by enabling caching of data for bulk encryption and decryption. We use s a base for the work presented in this paper.

11.3 The Secure Storage and Communication Framework

Our proposed secure storage and communication framework is based on the established IPv6/6LoWPAN protocols. IPv6/6LoWPAN defines IPsec/ESP (Encapsulating Security Payload) that provides encryption and authentication of transmitted data packets. We use the same cryptographic methods and data formats defined by ESP for data processing before storage. This requires us to store not only data but also all header information that is involved in the cryptographic processing. Encrypted data must be stored in ESP compatible form such that requested data can be transmitted over the network without further cryptographic processing. This requires us to anticipate content of communication protocol header fields such as IP destination addresses, sequence numbers and checksums at storage time. As IPsec is the base for communication and storage, the existing key exchange mechanisms defined for IPsec can be reused for the storage element of the framework.

The next subsection describes IPsec/ESP usage in 6LoWPAN networks. This represents the communication element of our framework. Thereafter follows a description of the storage element of the framework. We then briefly discuss application layer protocols that may be used with the framework and describe our Contiki based implementation. Finally we discuss expected performance gains and cost in terms of storage overhead and provide a security analysis.

11.3.1 Communication Component

IPv6 uses IPsec [2] to secure IP communication between two end points. IPsec is a collection of protocols that include Authentication Header (AH), which provides authentication services, and Encapsulating Security Payload (ESP), which provides both authentication and privacy services. A suite of encryption and authentication algorithms are also defined. A node keeps track of security associations (SA) that specify how IP flows are treated in terms of security.

In an ESP [18] packet data (for example, a UDP packet), padding, pad length and next header information are encrypted. All header information may be authenticated using the optional Integrity Check Value (ICV). In an 802.15.4 network an ESP header will not be transmitted directly. Its compressed form as defined by 6LoWPAN is used instead to reduce header overheads.

6LoWPAN defines header compression mechanisms. LOWPAN_IPHC is used for IP header compression and LOWPAN_NHC for the next header compression. The NH field in LOWPAN_IPHC when set to 1 indicates that the next header following the compressed IPv6 header is encoded with LOWPAN_NHC. LOWPAN_NHC has a length of 1 or more octets, where the first variable length bits identify the next header type and the remaining bits are used to encode header information. Currently, 6LoWPAN defines LOWPAN_NHC for the IP extension header (LOWPAN_NHC_EH) and the UDP header (LOWPAN_NHC_UDP). A definition for ESP encoding (LOWPAN_NHC_ESP) is provided in [3] and its fields are defined as:

- The first four bits in the LOWPAN_NHC_ESP represent the NHC ID defined for ESP. These are set to 1110.
- If $SPI = 00$: the default SPI for the 802.15.4 network is used and the SPI field is omitted. We set the default SPI value to 1. This does not mean that all nodes use the same security association (SA), but that every node has a single preferred SA, identified by SPI 1.

  If $SPI = 01$: First 8 bits of the SPI are carried inline; the remaining 24 bits are elided.

  If $SPI = 10$: First 16 bits of the SPI are carried inline; the remaining 16 bits are elided.

  If $SPI = 11$: All 32 bits of the SPI are carried inline.
- If $SN = 0$: The first 16 bits of sequence number are used. The remaining 16 bits are assumed to be zero.

  If $SN = 1$: All 32 bits of the sequence number are carried inline.
- If $NH = 0$: The next header field in ESP will be used to specify the next header and it is carried inline.

  If $NH = 1$: The next header will be encoded using LOWPAN_NHC. In case of ESP this would require the end systems to perform 6LoWPAN compression/decompression and encryption/decryption jointly.

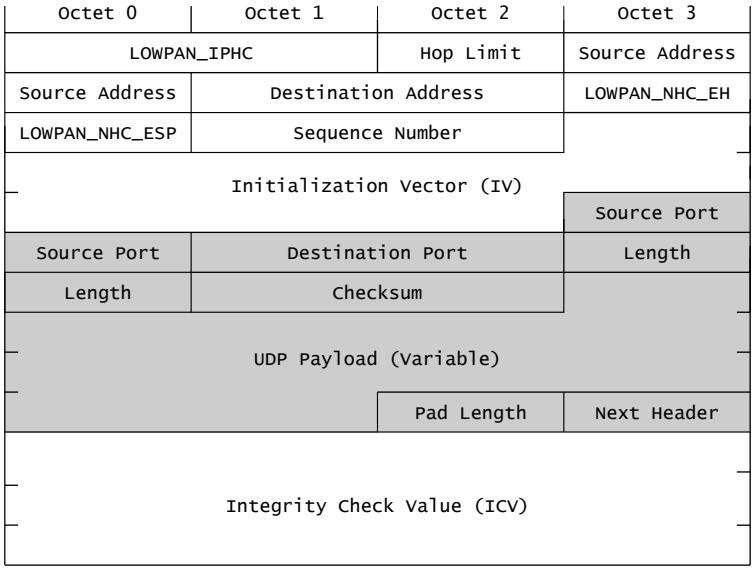


Figure 11.2: A compressed and ESP secured IPv6/UDP packet.

Figure 11.2 shows a UDP/IP packet secured with compressed ESP. An initialisation vector (IV) may be carried in the ESP packet if the selected encryption algorithm requires transmission of this information with every packet. The shaded portion represents encrypted data. Authentication can be provided using the ICV.

11.3.2 Storage Component

Data is stored securely such that it can be transmitted as ESP compliant packets on request without additional cryptographic processing. This requires storage of all cryptographically processed elements of the ESP packet within the file system. ESP header elements that are not cryptographically processed and can be constructed with little effort when data is requested and therefore do not have to be stored.

Data is stored as blocks representing the shaded part (and the ICV if authentication is required) shown in Figure 11.2. If data stored within a block is requested the block is read from the file system and the full packet, as shown in Figure 11.2, is assembled and transmitted. The receiver may only be interested

in part of the received data and some undesirable transmission overhead may occur. However, typical applications will require bulk data transfer (large parts of a file) in which case such overheads do not occur. For example, for further data analysis an application may request recorded sensor samples within a particular time frame or, for performance debugging purposes, recorded link quality metrics over a longer time period may be requested.

Some stored information is dependant on the communication relationship. At the time of storage, assumptions regarding the forthcoming communication relationship must be made in order to enable cryptographic processing. Elements to be considered are:

- *UDP Header:* A UDP header is stored within the encrypted ESP payload. Assumptions regarding destination and source UDP port must be made at time of storage. The destination IP address of packets is used within the IPv6 UDP checksum calculation. Thus, IP source and destination address assumptions must be made as well.

- *Initialisation Vector IV:* The IV (if required) is used for ESP encryption. Most protocols allow a counter mode where the IV for each packet is constructed by adding a transmission sequence number to an initial IV.

- *Sequence Number:* The ESP header includes a sequence number. This sequence number is not encrypted but it is included in the ICV calculation. If ESP authentication is used a sequence number must be selected at time of storage in order to generate a ICV for storage alongside the data.

UDP Header Construction: A UDP header has to be prepared at time of data storage. The header consists of 4 fields of $2byte$ length: Source Port, Destination Port, Length and Checksum.

The Length field is defined by the amount of data contained in the UDP packet. To reduce packet overheads the amount of data contained in each UDP packet is selected such that the maximum 802.15.4 frame size of $127byte$ is utilised.

The selection of a Source and Destination Port is not problematical. It can be assumed that well known ports can be used for data retreival.

The calculation of the Checksum field is challenging. The checksum is mandatory in IPv6 and is calculated using a pseudo header. This pseudo header contains the IP Source Address, the IP Destination Address, UDP Length and IP Next Header field. As the IP Destination Address is included assumptions

regarding the IP address requesting stored information must be made. The checksum is calculated as the 16-bit one's complement of the one's complement sum of the pseudo header, the UDP header, and the data.

It is a reasonable assumption that a particular host is used most of the time to request information from nodes (e.g. the sink). The IP address of this node may be used for storage preparation.

In some cases data may be requested from a different node and the IP destination address used for checksum calculation does not match the destination of the data requestor. In this situation it is possible to correct the checksum in a way that does not require the decryption and encryption of all of the data again. Thus, performance is reduced as part of the stored data must be cryptographically processed before transmission but it is still beneficial in comparison with a system that does not combine secure storage and transmission (See Evaluation).

ESP can use encryption algorithms which operate on blocks (e.g. AES using $16byte$ blocks). It is possible to decrypt only the first block of a larger stored ESP packet which will contain the UDP header and its checksum. Since the UDP checksum algorithm is a simple summation checksum re-calculation is trivial. By substituting the old destination address for the new destination address, a new checksum can be calculated. Now the first block of the ESP packet can be encrypted and it is ready for transmission to an alternative destination.

IV Construction: The IV does not have to be stored in the file system together with the encrypted ESP fields. An initial IV can be used and the storage block number is added to construct the IV.

Sequence Number Construction: If authentication is required it is possible to also store the ICV. As the ESP header includes a sequence number which is included in the ICV calculation, it is necessary to predict at storage time what sequence numbers will be used during communication.

Data belonging to a file is stored as sequence of ESP encrypted blocks and we can use the block number as ESP sequence number. IPsec allows us to reset the sequence number counters at the start of a communication relationship by establishing a new SA. Thereafter, data from the file can be delivered sequentially. In this setting we ensure that the communication uses sequence numbers that were selected at time of data storage.

11.3.3 Framework Usage

Application Layer Protocol: Nodes store data securely which may be requested by Internet hosts. Stored data has an application specific semantic. For exam-

ple, sensor values may be stored as a $4byte$ sensor value together with a $4byte$ time stamp and $2byte$ sequence number. Nodes execute a storage application that is able to respond to queries such as "send sensor samples recorded between 12:00:00 and 13:00:00". A host executes a storage application that is able to send these requests and to process arriving data. Host and node storage applications use UDP for communication. Similar to the well known FTP protocol, separate flows are used for command and data transfer which makes different IPsec security settings (including keys and security mechanisms) for both channels possible.

Security Configuration: The IPsec Security Association (SA) defines how data flows are protected. The SA holds secret keys, encryption algorithm descriptions and IP addresses to identify flows. Each SA holds a security parameters index (SPI), which is a 32-bit value used by a receiver to identify the correct SA.

If each file should be encrypted with a different key it is necessary to specify distinct SAs that each use a unique SPI. The SPI is transmitted in the 6LoWPAN header in compressed form (See Section 11.3.1). However, compression is only possible when the default SPI value is used; otherwise SPI information must be carried within the packet. Thus, the most frequently used file should use the default SPI in order to improve efficiency. In a practical setting this is not an issue as most nodes are using a single large file for storage of sensor data.

11.3.4 Implementation

We implemented the outlined framework for the Contiki [6] operating system. The implementation uses Contiki's μIP stack with 6LoWPAN/IPsec extensions as defined in [3] as the communication component. The storage component uses Contiki's Coffee filesystem (CFS) [7] with [5] to provide filesystem security extensions. The μIP stack was modified in order to enable direct passing of ESP encrypted packets from the filesystem to the communication stack. On the host side we used a standard Ubuntu Linux host.

For encryption/decryption we used AES in counter mode (CTR), with a $128bit$ key, in either hardware (e.g. via the CC2420 radio chip present on many sensor node platforms) or the MIRACL [19] library if hardware support is not available. If authentication is required, AES-XCBC-MAC-96 is used to calculate the necessary ICV (Provided via cryptographic processor or the MIRACL library).

The maximum 802.15.4 payload is 127byte and the available MAC layer

payload size is 102byte. As seen in Figure 11.2, 7byte are required for the compressed 6LoWPAN header, 12byte are required for the compressed ESP header fields, 2byte are required for the ESP trailer fields, 12byte are required for the ICV if it is used and 8byte are required for the UDP header. This leaves a maximum payload of 61byte. The AES algorithm requires a minimum block size of 16byte. Thus, the maximum feasible amount of data that can be stored per block before fragmentation must occur is 54byte. Storage blocks contain 64byte of encrypted data (8byte encrypted elements of the UDP header, 2byte encrypted ESP trailer and 54byte payload). Other feasible payload sizes are 6, 22 and 38. To avoid padding an application should align write operations with these payload sizes.

At this point in time, our Contiki IPsec implementation does not support key exchange mechanisms such as the Internet Key Exchange (IKE) protocol. Keys are set manually before deployment. However, for most application scenarios this would not be an issue limiting the frameworks usability.

11.3.5 Security Discussions

In this section we briefly discuss the security of the combined storage and communication system. We consider key management, cryptographic algorithms, message encryption and message authentication. In particular, we determine if the combination of secure storage and secure communication provides weaker security than systems treating both subsystems individually.

Confidentiality - Communication is secured using IPsec's ESP procedures. The solution does not deviate from procedures defined in the IPsec framework. An attacker with access to the communication channel has access to the same information as an attacker on any other ESP secured communication. If we consider IPsec a secure solution the provided solution can be considered secure as well.

Our implementation uses AES in counter mode (CTR) with $128bit$ keys. The best known AES attack for this key length is four times better than exhaustive search [20], and does not adversely affect its security.

Integrity and Authentication - When authentication is required, the ICV is calculated and appended to the ESP. Here we have to balance security and performance needs. Storing the ICV along with the ESP will ensure data integrity and authentication for storage and communication. However, when storing ICVs along with the encrypted payload it is necessary to select sequence numbers at the time data is stored. Hence, sequence numbers are predictable and will repeat when stored file content is transmitted repeatedly. Thus, protection

against replay attacks in the communication channel is weakened. On the other hand, we will have performance gains as the ICV does not have to be computed at transmission time. If we decide to calculate the ICV before each transmission the replay protection is strongly enforced while the performance gains are reduced and stored data is missing integrity and authentication data.

To provide both, strong data integrity and relatively weaker anti-replay, functionalies when the ESP authentication field (ICV) is also stored in flash memory, sequence numbers should be in order before calculating ICVs for all the stored packets in a file, and the sender's and receiver's counter should be reset (by establishing a new SA) prior to the transmission of a file.

Storage - Data is stored in the same format as it is later transmitted. An attacker with access to the file system has the same information available as an attacker with access to the communication channel. If transmitted information secured using ESP is considered to be secure then information stored in the file system must be considered secure as well.

Key Management - Data in flash memory is secured using the same key that is later used for communication. Hence, transmission of the same stored data requires usage of the same key on the communication link. It is not possible to negotiate a fresh key for each communication relationship compared to when IPsec is used on its own. However, many practical IPsec deployments use pre-shared fixed keys so we consider this a secure option.

Similarly, if multiple nodes have to be able to access the same stored information they will also have to use the same key for communication. This is similar to practical situations where IPsec is used with a single pre-shared key.

The proposed system has difficulties with revocation of keys; if a new key is selected data already stored in the flash file system must be re-encrypted, which is costly on resource constrained systems.

11.4 Evaluation

In this section we first discuss the costs in terms of storage overhead that are associated with the proposed scheme of combined storage and communication. Thereafter we analyse the processing performance and energy consumption gains associated with our scheme. We use our Contiki implementation for the Telos B platform.

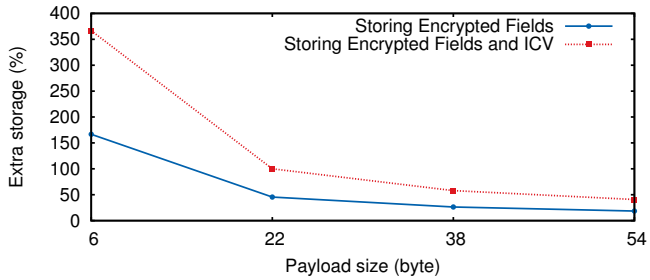


Figure 11.3: Storage overheads for different payload sizes.

11.4.1 Storage Overheads

Storing encrypted data together with the ESP fields that demand cryptographic processing requires additional storage space compared to a solution which would only store encrypted data.

If only encryption is used an extra $10byte$ per stored data block is required. Thus, it is better to store large blocks (large ESP packets) as this reduces the overhead. Figure 11.3 shows the overhead in dependency of the payload size. We show overheads for payload sizes which align with the AES encryption block size of $16byte$ and that do not require fragmentation when transmitted.

If authentication is also required then overheads increase as the ICV data of $12byte$ has to be stored alongside the other encrypted information.

The results show that the proposed framework reduces the effective storage size of the available flash storage space on nodes by 40.7% when using a payload size of $54byte$ and use of ICV. However, if we consider a common flash memory size of $16GB$ in which an $10byte$ sensor reading is recorded every minute the time until the storage capacity is exceeded is reduced from $3266years$ to $1329years$. Both values are acceptable in any deployment context and it can be concluded that the necessary storage overhead is not a limiting factor of the proposed framework.

11.4.2 Performance Gains

The combined storage and communication framework provides performance improvements. To analyse performance benefits in detail we use 4 different experiments. In all 4 experiments, ESP encryption and authentication is provided. The different experiments are used to show increasing performance

benefits with increasing integration of storage and communication. *Experiment A* uses a system without the combining storage and communication. In these experiments (*Experiment B, C, D*) the framework is used in different configurations.

Experiment A is the baseline experiment where data is read from flash memory, decrypted and re-encrypted for IPsec conform transmission. In addition, authentication is provided and an ESP ICV is constructed. In *Experiment B*, ESP encrypted fields are stored in the flash memory and can be directly transmitted upon request. The ICV for authentication is still constructed at transmission time. *Experiment C* differs from *Experiment B* in terms of UDP checksum calculation. A non-matching IP address was used at storage time and a re-calculated before transmission is necessary. In *Experiment D*, ESP encrypted fields and the ESP authentication field (ICV) are stored in flash memory and can be transmitted directly upon request. All of the experiments are carried out with both software and hardware encryption. Table 11.1 summarises the experiment settings.

	Encryption	Authentication	UDP checksum re-calculation
Experiment A	individual	individual	-
Experiment B	combined	individual	not required
Experiment C	combined	individual	required
Experiment D	combined	combined	not required

Table 11.1: Experiment setup details used for evaluation. All experiments use ESP encryption and authentication. The combined storage and communication framework is used for different aspects. UDP checksum re-calculation is assumed in some settings.

Experiment A: Baseline experiment

In this experiment, data is read from a file and sent using conventional methods. ESP conform AES encryption is used for the storage component and communication component to allow for comparison with the other experiments. Payload data is read in blocks of 6, 22, 38 and $54bytes$. The payload is decrypted and then re-encrypted for IPsec transmission. ICV authentication data is constructed before transmission. Decryption is carried out within the Contiki CFS; encryption and ICV calculation is carried out within the Contiki μIP stack.

Figure 11.6 shows the time that is necessary on a node to process one payload. The processing time is measured from the start of the file system read operation to the completion of the packet transmission. The total processing time is broken down to show the contribution of significant individual operations:

- *CFS reading* is the time required to read data from the file system.
- *CFS decryption* represents the time necessary to perform data decryption.
- *ESP encryption* represents the time necessary to encrypt the ESP payload.
- *ESP ICV calculation* is the time required to produce authentication data.
- *Other operations* summarises the duration of all other operations.

Figure 11.4 shows the processing duration breakdown when cryptographic processing is carried out in software. The total time to prepare a single packet is $19.1ms$, $25.2ms$, $31.4ms$ and $37.6ms$ for $6byte$, $22byte$, $38byte$ and $54byte$ payload data, respectively. CFS reading time is 8%, CFS decryption time is 21.3%, ESP encryption time is 21.5% and ESP ICV calculation time is 25.1% of the overall processing time for $54byte$ payload.

Figure 11.5 shows the duration of operations when using hardware supported cryptographic processing. Total times for preparing a single packet are $11.8ms$, $14.5ms$, $17.1ms$ and $19.7ms$ for the different payload sizes. CFS reading time is 15.1%, CFS decryption time is 12.5%, ESP encryption time is 12.7% and ESP ICV calculation time is 13.8% of the overall time when preparing $54byte$ of data.

Enabling hardware support improves performance by 38.1%, 42.6%, 45.5% and 47.5% for $6byte$, $22byte$, $38byte$ and $54byte$ payloads, respectively.

The experiments show that when a $54byte$ payload is transmitted processing the node spends 67.9% of the preparation time on cryptographic processing (software supported). This cryptographic processing time can be avoided by the proposed framework as we show in the following experiments.

Experiment B: Storing ESP fields

In this experiment, ESP encrypted fields are stored in flash memory with the payload data. 6, 22, 38 and $54byte$ payloads are used. As additional stored

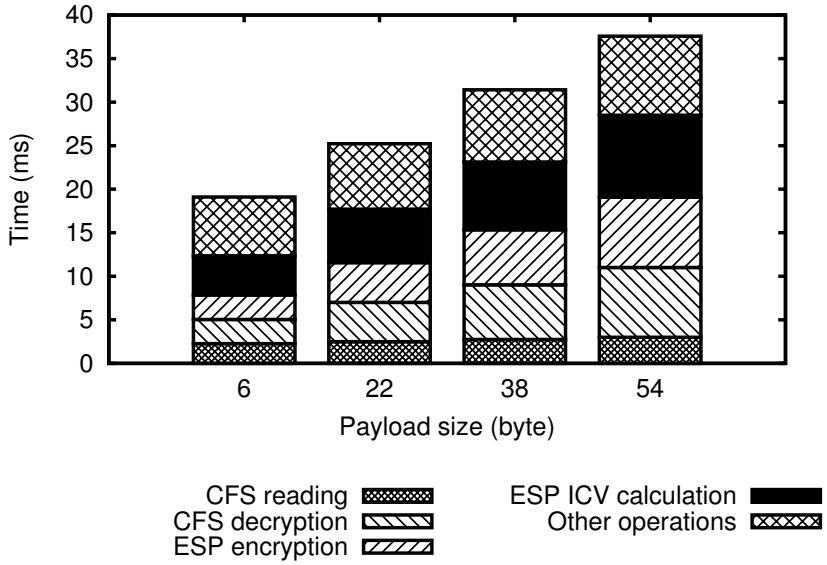


Figure 11.4: With software encryption.

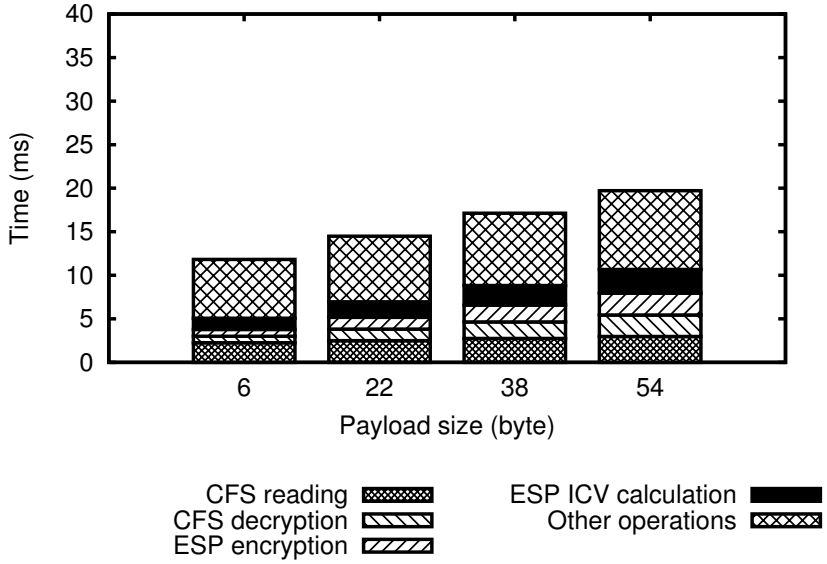


Figure 11.5: With hardware encryption.

Figure 11.6: Duration of different operations involved in preparing single packet for transmission with software and hardware encryption.

ESP fields are $10byte$ length $16byte$, $32byte$, $48byte$ and $64byte$ must be read from the file system. Compared to *Experiment A* CFS decryption and ESP encryption is now not necessary, so processing time is saved.

Figure 11.7 shows the duration for different operations when preparing a single packet for transmission with software encryption. Total times for preparing a single packet are $12.5ms$, $15ms$, $17.6ms$ and $20.2ms$; and improvements in system performance when it is compared to the baseline experiment with software encryption are 34.6%, 40.4%, 43.9% and 46.3%.

Figure 11.8 shows processing times when using hardware encryption. Total times for preparing a single packet are $9.3ms$, $10.7ms$, $12.1ms$ and $13.5ms$; and improvements in system performance when it is compared to the baseline experiment with software encryption are 51.3%, 57.7%, 61.6% and 64.1%.

It is notable that the CFS read time is less than in *Experiment A* even though more data has to be read from the file system (e.g. instead of $54byte$, $64byte$ are read as ESP information is included). This is due to the fact that elements of the CFS can be bypassed when directly reading encrypted data for transmission.

Experiment C: Using a non-matching IP address

This experiment is similar to *Experiment B*. The difference is the IP address of the destination when carrying out ESP encryption for storage. The UDP checksum enclosed in ESP packets must be corrected before transmission. The time necessary to perform decryption of the $16byte$ block containing the checksum, its correction and encryption of the $16byte$ block containing the corrected checksum is referred to as *UDP checksum preparation*.

Figure 11.10 shows results using software encryption. The total time for preparing a single packet is $15.3ms$, $17.9ms$, $20.4ms$ and $23ms$; and improvements in system performance when it is compared to the baseline experiment with software encryption are 20.1%, 29.1%, 35.1% and 38.9% for the different payload sizes.

Figure 11.11 shows results when using cryptographic hardware support. Total times are $10ms$, $11.4ms$, $12.8ms$ and $14.2ms$; and improvements in system performance when it is compared to the baseline experiment with software encryption are 47.6%, 54.7%, 59.2% and 62.1% for 6, 22, 38 and $54byte$ data.

The correction of the UDP checksum, which may be necessary in cases we cannot anticipate the endpoint to which stored data must be delivered, is not very costly. For a $54byte$ payload using hardware support the performance gain is only reduced from 64.1% to 62.1%.

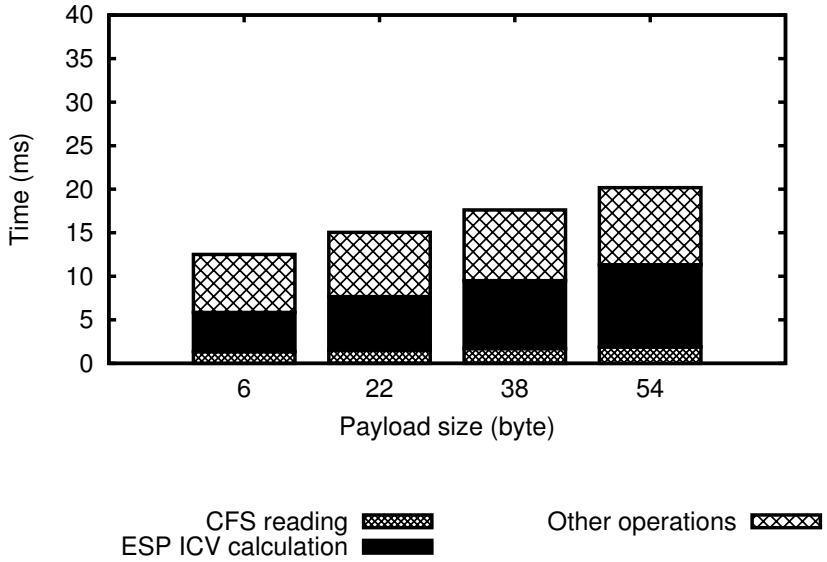


Figure 11.7: With software encryption.

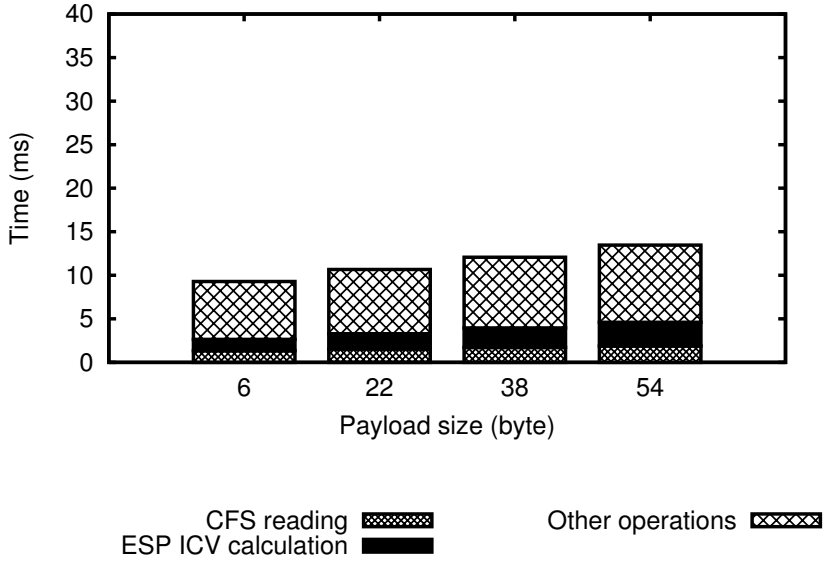


Figure 11.8: With hardware encryption.

Figure 11.9: Duration of different operations involved in preparing single packet for transmission with software and hardware encryption when storing ESP encrypted fields.

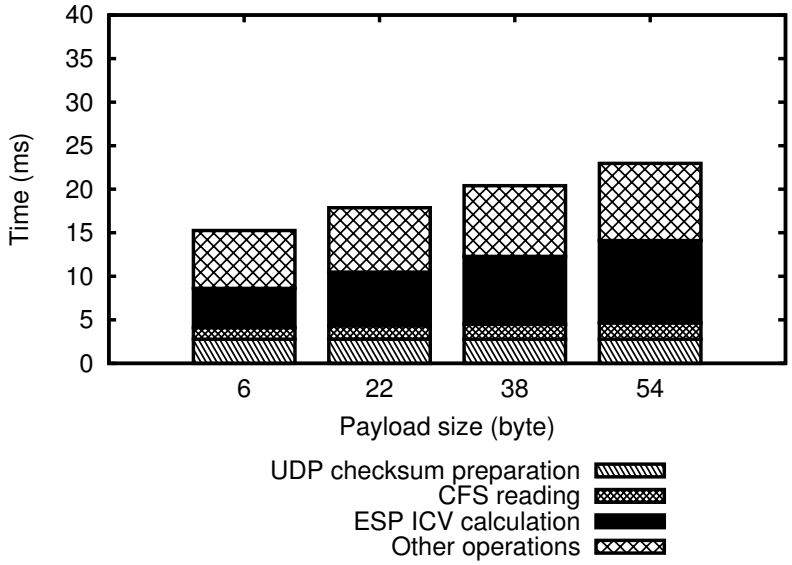


Figure 11.10: With software encryption.

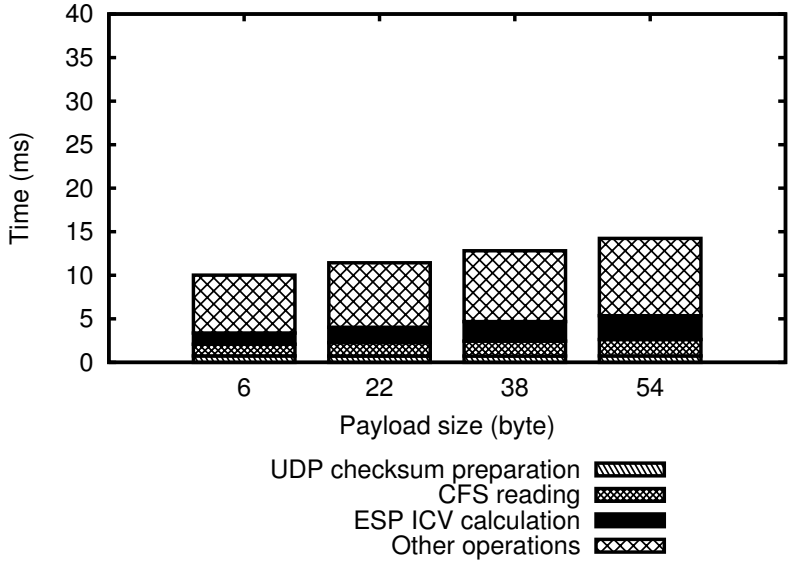


Figure 11.11: With hardware encryption.

Figure 11.12: Duration of different operations involved in preparing single packet for transmission with software and hardware encryption when using a non-matching IP address.

Experiment D: Storing ESP and ICV fields

In this experiment all options of the proposed framework are in use. Data is stored in ESP compatible form alongside the ICV authentication data. In this case no encryption processing is required when data is requested, and thus processing times are independent from cryptographic algorithm implementations (hardware or software). The results are shown in Figure 11.13. For direct comparison we again show the results of *Experiment A* (software encryption) in Figure 11.14.

In this experiment, ESP encrypted fields and ESP authentication field (ICV) are stored in flash memory together with the payload data. As encrypted fields have a length of $10bytes$ and the authentication field (ICV) is $12bytes$ long, blocks of $28byte$, $44byte$, $60byte$ and $76byte$ have to be read for different payload sizes.

Compared to the previous two experiments, CFS read times increase as the additional ICV has to be read. Total time for preparing a single packet is $8.1ms$, $9.1ms$, $10ms$ and $10.9ms$. Improvements in system performance when compared with the baseline experiment with software encryption are 57.5%, 64.1%, 68.3% and 71% for 6, 22, 38 and $54byte$ payload data.

These results show that the proposed framework of combined storage and communication can achieve significant performance gains. When the framework is used, requested data can be delivered approximately 3 times faster as cryptographic processing is not required at the time when data is prepared for delivery.

11.4.3 Energy Consumption

We have shown that combining secure storage and communication reduces processing time on sensor nodes. However, it is not immediatly clear if savings in processing time translate to energy savings as the proposed mechanism changes usage patterns of hardware components such as flash memory and hardware encryption.

We therefore compare energy consumption of the conventional storage method with our combined storage and communication method. We use the setups previously described as *Experiment A* and *Experiment B*. We use energy consumption values for CC2420 radio operations and ST M25P80 flash operations from the Tmote Sky datasheet [21], and for CC2420 hardware cryptographic support from [22].

If $54byte$ data is required to be stored and transmitted later using the con-

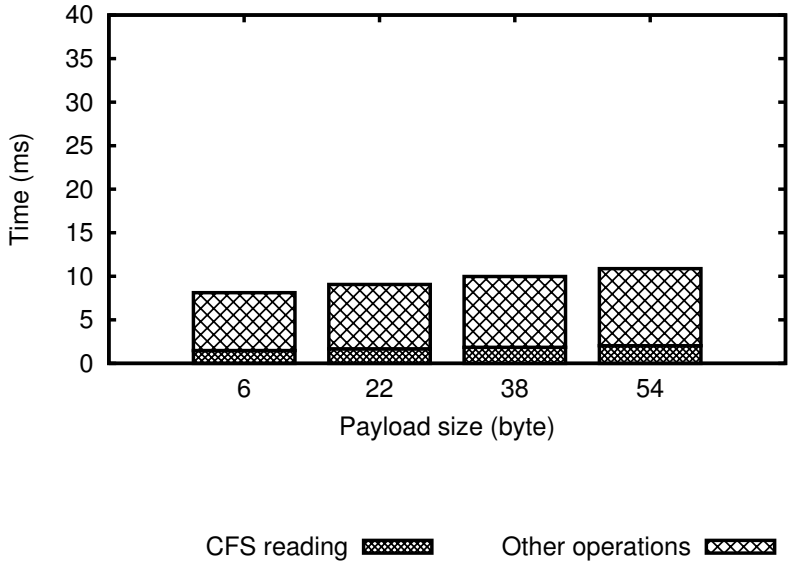


Figure 11.13: Combined storage and communication

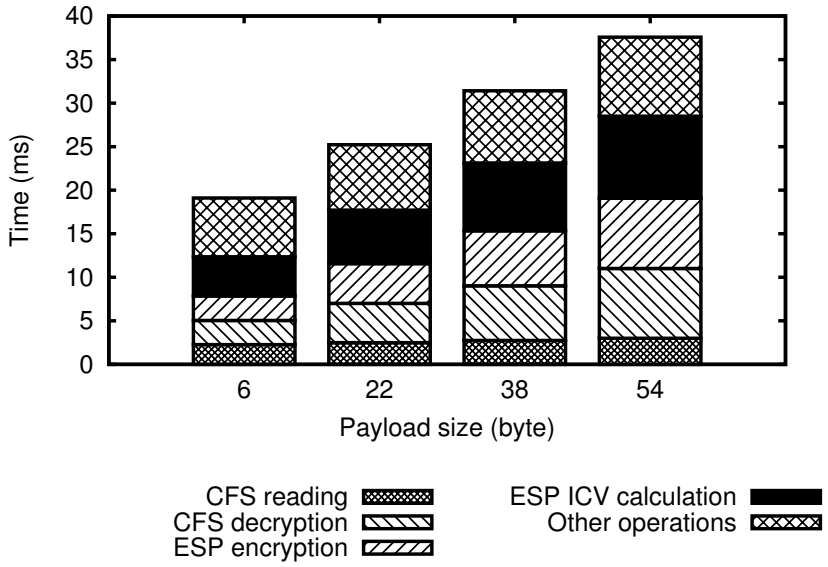


Figure 11.14: Individual security processing

Figure 11.15: Duration of different operations necessary to prepare a single packet for transmission when using the combined storage and communication framework and when using individual storage and communication security solutions.

ventional method, $54byte$ data has to be encrypted and written to the flash memory for storage; $54byte$ data has to be read from the flash memory and has to be decrypted; $64byte$ data has to be encrypted in IPsec; $80byte$ data has to be authenticated in IPsec and the packet has to be transmitted, respectively. In case that $54byte$ of data is required to be stored and transmitted using combined storage and communication method, $64byte$ data has to be encrypted and written to flash memory for storage; $64byte$ data has to be read from the flash memory; $80byte$ of data has to be authenticated and the packet has to be transmitted.

The system is able to skip two cryptographic operations when using combined secure storage and communication. Therefore, the energy consumption decreases by 32.1%, even when additional $10byte$ have to be written to and read from flash memory. Due to space restrictions we do not detail energy savings for all other experiment combinations as discussed in the previous section. However, in all cases our proposed method leads to energy savings. In the worst-case, energy consumption decreases by 18.7% (in *Experiment D* with $6byte$ data size).

11.5 Conclusion

We have shown that combined secure storage and communication can reduce security related real-time processing on nodes dramatically (up to 71% reduction). As shown, this can be achieved while decreasing as well a nodes power consumption (up to 32.1%). Furthermore, we have shown that this is possible within the context of the IP protocol family which we believe will be used in the future IoT. The described solution requires additional storage space on nodes. However, we believe that currently available flash memory sizes can absorb these overheads.

Data on nodes must be secured when stored *and* transported in order to implement a comprehensive security solution. As resource-constrained embedded systems are limited in resources it is necessary to find efficient solutions. As shown, the proposed framework combining security aspects of storage and communication can help to achieve this goal.

11.6 Acknowledgements

This work has been partially supported by SSF.

Bibliography

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

[2] S. Kent and K. Seo. Security Architecture for the Internet Protocol. RFC 4301, 2005.

[3] Shahid Raza, Simon Duquennoy, Joel Hglund, Utz Roedig, and Thiemo Voigt. Secure communication for the internet of things - a comparison of link-layer security and ipsec for 6lowpan. *Security and Communication Networks*, 2012.

[4] Nicolas Tsiftes and Adam Dunkels. A database in every sensor. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, pages 316–332. ACM, 2011.

[5] Ibrahim Ethem Bagci, Mohammad Reza Pourmirza, Shahid Raza, Utz Roedig, and Thiemo Voigt. Codo: Confidential data storage for wireless sensor networks. In *8th IEEE International Workshop on Wireless and Sensor Networks Security (WSNS 2012)*, Las Vegas, Nevada, USA, October 2012.

[6] Adam Dunkels, Bjorn Gronvall, and Thiemo Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks*, pages 455–462. IEEE Computer Society, 2004.

[7] Nicolas Tsiftes, Adam Dunkels, He Zhitao, and Thiemo Voigt. Enabling large-scale storage in sensor networks with the coffee file system. In *Proceedings of the 2009 International Conference on Information Processing in Sensor Networks*, pages 349–360, Washington, DC, USA, 2009. IEEE Computer Society.

[8] Piotr Szczechowiak, Leonardo B. Oliveira, Michael Scott, Martin Collier, and Ricardo Dahab. Nanoecc: testing the limits of elliptic curve cryptography in sensor networks. In *Proceedings of the 5th European conference on Wireless sensor networks*, pages 305–320. Springer-Verlag, 2008.

[9] Wen Hu, Peter Corke, Wen Chan Shih, and Leslie Overs. secfleck: A public key technology platform for wireless sensor networks. In *Proceedings of the 6th European Conference on Wireless Sensor Networks*, pages 296–311. Springer-Verlag, 2009.

[10] Donggang Liu and Peng Ning. Establishing pairwise keys in distributed sensor networks. In *Proceedings of the 10th ACM conference on Computer and communications security*, CCS '03, pages 52–61. ACM, 2003.

[11] Neerja Bhatnagar and Ethan L. Miller. Designing a secure reliable file system for sensor networks. In *Proceedings of the 2007 ACM workshop on Storage security and survivability*, pages 19–24, 2007.

[12] IEEE std. 802.15.4 - 2003. Wireless medium access control (mac) and physical layer (phy) specifications for low rate wireless personal area networks (lr-wpans). *IEEE*, 2003.

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

[14] Sungmin Hong, Daeyoung Kim, Minkeun Ha, Sungho Bae, Sang Jun Park, Wooyoung Jung, and Jae-Eon Kim. Snail: an ip-based wireless sensor network approach to the internet of things. *Wireless Communications, IEEE*, 17(6):34 –42, december 2010.

[15] Sepideh Fouladgar, Bastien Mainaud, Khaled Masmoudi, and Hossam Afifi. Tiny 3-tls: a trust delegation protocol for wireless sensor networks. In *Proceedings of the Third European conference on Security and Privacy in Ad-Hoc and Sensor Networks*, pages 32–42. Springer-Verlag, 2006.

[16] Joao Girao, Dirk Westhoff, Einar Mykletun, and Toshinori Araki. Tinypeds: Tiny persistent encrypted data storage in asynchronous wireless sensor networks. *Ad Hoc Netw.*, 5:1073–1089, September 2007.

[17] Wei Ren, Yi Ren, and Hui Zhang. Hybrids: A scheme for secure distributed data storage in wsns. In *Proceedings of the 2008 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing - Volume 02*, pages 318–323. IEEE Computer Society, 2008.

[18] S. Kent. IP Encapsulating Security Payload (ESP). RFC 4303, 2005.

[19] CertiVox. MIRACL - Multiprecision Integer and Rational Arithmetic C/C++ Library.

[20] Andrey Bogdanov, Dmitry Khovratovich, and Christian Rechberger. Biclique cryptanalysis of the full aes. In *Proceedings of the 17th international conference on The Theory and Application of Cryptology and Information Security*, pages 344–371. Springer-Verlag, 2011.

[21] Tmote Sky datasheet, 2006. `http://www.eecs.harvard.edu/~konrad/projects/shimmer/references/tmote-sky-datasheet.pdf`.

[22] Fan Zhang, Reiner Dojen, and Tom Coffey. Comparative performance and energy consumption analysis of different aes implementations on a wireless sensor network node. *International Journal of Sensor Networks*, 10(4):192–201, 2011.

Swedish Institute of Computer Science

SICS Dissertation Series

01: Bogumil Hausman, Pruning and Speculative Work in OR-Parallel PROLOG, 1990.

02: Mats Carlsson, Design and Implementation of an OR-Parallel Prolog Engine, 1990.

03: Nabiel A. Elshiewy, Robust Coordinated Reactive Computing in SANDRA, 1990.

04: Dan Sahlin, An Automatic Partial Evaluator for Full Prolog, 1991.

05: Hans A. Hansson, Time and Probability in Formal Design of Distributed Systems, 1991.

06: Peter Sjödin, From LOTOS Specifications to Distributed Implementations, 1991.

07: Roland Karlsson, A High Performance OR-parallel Prolog System, 1992.

08: Erik Hagersten, Toward Scalable Cache Only Memory Architectures, 1992.

09: Lars-Henrik Eriksson, Finitary Partial Inductive Definitions and General Logic, 1993.

10: Mats Björkman, Architectures for High Performance Communication, 1993.

11: Stephen Pink, Measurement, Implementation, and Optimization of Internet Protocols, 1993.

12: Martin Aronsson, GCLA. The Design, Use, and Implementation of a Program Development System, 1993.

13: Christer Samuelsson, Fast Natural-Language Parsing Using Explanation-Based Learning, 1994.

14: Sverker Jansson, AKL - - A Multiparadigm Programming Language, 1994.

15: Fredrik Orava, On the Formal Analysis of Telecommunication Protocols, 1994.

16: Torbjörn Keisu, Tree Constraints, 1994.

17: Olof Hagsand, Computer and Communication Support for Interactive Distributed Applications, 1995.

18: Björn Carlsson, Compiling and Executing Finite Domain Constraints, 1995.

19: Per Kreuger, Computational Issues in Calculi of Partial Inductive Definitions, 1995.

20: Annika Waern, Recognising Human Plans: Issues for Plan Recognition in Human-Computer Interaction, 1996.

21: Björn Gambäck, Processing Swedish Sentences: A Unification-Based Grammar and Some Applications, 1997.

22: Klas Orsvärn, Knowledge Modelling with Libraries of Task Decomposition Methods, 1996.

23: Kia Höök, A Glass Box Approach to Adaptive Hypermedia, 1996.

24: Bengt Ahlgren, Improving Computer Communication Performance by Reducing Memory Bandwidth Consumption, 1997.

25: Johan Montelius, Exploiting Fine-grain Parallelism in Concurrent Constraint Languages, 1997.

26: Jussi Karlgren, Stylistic experiments in information retrieval, 2000.

27: Ashley Saulsbury, Attacking Latency Bottlenecks in Distributed Shared Memory Systems, 1999.

28: Kristian Simsarian, Toward Human Robot Collaboration, 2000.

29: Lars-Åke Fredlund, A Framework for Reasoning about Erlang Code, 2001.

30: Thiemo Voigt, Architectures for Service Differentiation in Overloaded Internet Servers, 2002.

31: Fredrik Espinoza, Individual Service Provisioning, 2003.

32: Lars Rasmusson, Network capacity sharing with QoS as a financial derivative pricing problem: algorithms and network design, 2002.

33: Martin Svensson, Defining, Designing and Evaluating Social Navigation, 2003.

34: Joe Armstrong, Making reliable distributed systems in the presence of software errors, 2003.

35: Emmanuel Frécon, DIVE on the Internet, 2004.

36: Rickard Cöster, Algorithms and Representations for Personalised Information Access, 2005.

37: Per Brand, The Design Philosophy of Distributed Programming Systems: the Mozart Experience, 2005.

38: Sameh El-Ansary, Designs and Analyses in Structured Peer-to-Peer Systems, 2005.

39: Erik Klintskog, Generic Distribution Support for Programming Systems, 2005.

40: Markus Bylund, A Design Rationale for Pervasive Computing User Experience, Contextual Change, and Technical Requirements, 2005.

41: Åsa Rudström, Co-Construction of hybrid spaces, 2005.

42: Babak Sadighi Firozabadi, Decentralised Privilege Management for Access Control, 2005.

43: Marie Sjölinder, Age-related Cognitive Decline and Navigation in Electronic Environments, 2006.

44: Magnus Sahlgren, The Word-Space Model: Using Distributional Analysis to Represent Syntagmatic and Paradigmatic Relations between Words in High-dimensional Vector Spaces, 2006.

45: Ali Ghodsi, Distributed k-ary System: Algorithms for Distributed Hash Tables, 2006.

46: Stina Nylander, Design and Implementation of Multi-Device Services, 2007.

47: Adam Dunkels, Programming Memory-Constrained Networked Embedded Systems, 2007.

48: Jarmo Laaksolahti, Plot, Spectacle, and Experience: Contributions to the Design and Evaluation of Interactive Storytelling, 2008.

49: Daniel Gillblad, On Practical Machine Learning and Data Analysis, 2008.

50: Fredrik Olsson, Bootstrapping Named Entity Annotation by Means of Active Machine Learning: a Method for Creating Corpora, 2008.

51: Ian Marsh, Quality Aspects of Internet Telephony, 2009.

52: Markus Bohlin, A Study of Combinatorial Optimization Problems in Industrial Computer Systems, 2009.

53: Petra Sundström, Designing Affective Loop Experiences, 2010.

54: Anders Gunnar, Aspects of Proactive Traffic Engineering in IP Networks, 2011.

55: Preben Hansen, Task-based Information Seeking and Retrieval in the Patent Domain: Process and Relationships, 2011.

56: Fredrik Österlind, Improving Low-Power Wireless Protocols with Timing-Accurate Simulation, 2011.

57: Ahmad Al-Shishtawy, Self-Management for Large-Scale Distributed Systems, 2012.

58: Henrik Abrahamsson, Network Overload Avoidance by Traffic Engineering and Content Caching, 2012.

59: Mattias Rost, Mobility is the Message: Experiment with Mobile Media Sharing, 2013

60: Amir H. Payberah, Live Streaming in P2P and Hybrid P2P-Cloud Environments for the open Internet, 2013

61: Oscar Täckstrm, Predicting Linguistic Structure with Incomplete and Cross-Lingual Supervision, 2013

62: Cosmin Arad, Programming Model and Protocols for Reconfigurable Distributed Systems, 2013

63: Tallat M. Shafaat, Partition Tolerance and Data Consistency in Structured Overlay Networks, 2013

64: Shahid Raza, Lightweight Security Solutions for the Internet of Things, 2013