

Performance Evaluation of end-to-end security protocols in an Internet of Things

Antonio De Rubertis, Luca Mainetti, Vincenzo
Mighali, Luigi Patrono, Ilaria Sergi, Maria Laura
Stefanizzi
Dept. of Innovation Engineering
University of Salento
Lecce, ITALY

Stefano Pascali
STMicroelectronics
Advanced System Technology
Lecce, ITALY

Abstract— Wireless Sensor Networks are destined to play a fundamental role in the next-generation Internet, which will be characterized by the Machine-to-Machine paradigm, according to which, embedded devices will actively exchange information, thus enabling the development of innovative applications. It will contribute to assert the concept of Internet of Things, where end-to-end security represents a key issue. In such context, it is very important to understand which protocols are able to provide the right level of security without burdening the limited resources of constrained networks. This paper presents a performance comparison between two of the most widely used security protocols: IPSec and DTLS. We provide the analysis of their impact on the resources of embedded devices. For this purpose, we have modified existing implementations of both protocols to make them properly run on our hardware platforms, and we have performed an extensive experimental evaluation study. The achieved results are not a consequence of a classical simulation campaign, but they have been obtained in a real scenario that uses software and hardware typical of the current technological developments. Therefore, they can help network designers to identify the most appropriate secure mechanism for end-to-end IP communications involving constrained devices.

Keywords—Security; IPSec; DTLS; Performance evaluation; WSN; Contiki RTOS; IoT; Test bed.

I. INTRODUCTION

Over the last few years, Wireless Sensor Networks (WSNs), and more in general the Low-Rate Wireless Personal Area Network (i.e., those networks compliant with IEEE 802.15.4 protocol), are becoming more and more important in the field of smart technologies [1]. The opportunity to realize pervasive environments that can detect environmental parameters in a precise and efficient way is becoming more attractive for both the academic and the industrial world. Therefore, WSNs are widely used in various scenarios and in many applications, ranging from environmental monitoring, to intrusion detection, to chemical and biological attacks prevention, to health monitoring, etc. Their use is justified by their peculiar characteristics, i.e., the ability to detect key parameters from the surrounding environment, the small size of physical devices, the low-cost, the ability to self-configure and self-organize, the ability to communicate through the wireless channel, etc. However, as can be deduced from the short list of

possible applications previously described, in some scenarios several stringent requirements in terms of security and privacy may be needed. If the network is not adequately protected an attacker could generate unsafe situations (e.g., stealing sensitive data). In recent times, the development of specific protocols for constrained networks, such as 6LoWPAN [2] and CoAP [3], has made WSNs easily accessible via the public Internet, thus favoring the assertion of the Internet of Things (IoT) concept, but, at the same time, exposing them to new security threats. In accordance with the IoT vision, every-day objects will become proactive actors of the global Internet, with the capability of generating and consuming information for advanced applications. In this way, Human-to-Machine (H2M) paradigm is increasingly moving toward the new Machine-to-Machine (M2M) paradigm, so leading to an improvement of several aspects in everyday life. However, these considerable advantages in terms of availability of resources are often associated with possible external attacks, which require a greater awareness regarding the safety aspect in network communications. Although the IEEE 802.15.4 standard provides and documents the security mechanisms at the MAC layer (i.e., within the WSN), nowadays there is no standard protocol or suite of standard protocols that ensures the end-to-end security at network and/or higher layers. Therefore, this topic is of great interest to the scientific community.

The implementation of protocols in constrained networks has to deal with some problems related to the particular nature of the physical devices. The limited computational capacity, the low amount of memory, and the constraints on the energy consumption, make the design of these protocols particularly hard and complicated [4, 5] and, in some cases, they suggest the use of hardware components that do not impact on the limited resources of constrained devices [6, 7]. This problem is much more evident for security protocols, which need minimum computational requirements that exceed the CPU capability of constrained devices and introduce an excessive amount of overhead compared to the maximum packet size allowed by the standard. In the literature, there are some works focused on the adaptation of standard security protocols to constrained networks. In [8], the authors propose and evaluate new compressed Authentication Header (AH) and Encapsulation Security Payload (ESP) of 6LoWPAN that allow the protection of IPv6 communications on IEEE 802.15.4

WSNs. As the security headers are designed to work side-by-side with other extension headers of the 6LoWPAN adaptation layer, they allow also the establishment of end-to-end secure communications between Internet hosts and wireless sensor nodes. Another work on the secure communications in a constrained network is focused on DTLS [9]. Since DTLS is a heavyweight protocol and its headers are too long to fit in a single IEEE 802.15.4 MTU, the authors propose 6LoWPAN header compression for DTLS and show that this compression significantly reduces the number of additional security bits. Finally, in [10], an implementation of TLS/DTLS protocol, which runs over the 6LoWPAN IPv6 adaptation layer in the Contiki OS [11], is presented.

Key goals of this work were to evaluate the effectiveness of the usage of secure end-to-end communications in the context of Internet-integrated sensing applications, and to analyze if current sensing platforms are able to cope with current requests for security. In particular, IPsec and DTLS security protocols have been implemented on the available physical devices and deeply evaluated in terms of feasibility, performance and cost. The obtained results showed that the porting of both protocols is able to meet the stringent requirements of a constrained network and neither of the two protocols clearly overcomes the other. Such results allowed us to assert that an optimal security solution for any applicative scenario does not exist, but the use of one protocol with respect to the other depends on the contingent situation, that is, on the requirements and the application fields in which embedded devices are used.

The rest of the paper is organized as follows. Section II gives an overview on the security protocols used in our performance evaluation. Implementation details are presented in Section III, whereas in Section IV the test environment is deeply described. Section V presents the obtained results in terms of both performance and resources usage. Conclusions are drawn in Section VI.

II. OVERVIEW ON STANDARD SECURITY PROTOCOLS

In this section, a brief overview of core functionalities of IPsec and DTLS that are relevant for the work presented in this paper is provided.

A. IPsec protocol suite: IKE, AH, and ESP

IPsec [12] defines a protocols suite for securing IP communication on an end-to-end basis, including: the security protocols AH [13] and ESP [14], the algorithms for authentication and encryption, and key exchange mechanisms. AH and ESP do not handle the keys exchange, since they assume that the two nodes have already created a Security Association (SA) [12]. A SA is a "contract" between the two IPsec endpoints used to establish the protection mechanisms and the keys to be used during the next data transfer. For this purpose, IPsec standard specifies both Pre-Shared Key (PSK) mechanism and Internet Key Exchange (IKE) [15] protocol. However, the latter uses asymmetric cryptography that is assumed to be heavy weight for small sensor nodes.

Both AH and ESP protocols support connectionless integrity, anti-replay protection, and data origin authentication. AH authenticates the whole IP packet, with the exception of

the IP header variable fields which, being modified by intermediate nodes, cannot be authenticated. Unlike AH, ESP supports confidentiality as well. ESP is used to encrypt the payload of an IP packet but in contrast to AH it does not secure the IP header.

The AH and ESP protocols provide two operating modes: transport mode and tunnel mode. According to the first one, IP header and payload are encrypted 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 IP packet.

B. Datagram Transport Layer Security

The Datagram Transport Layer Security (DTLS) [16] protocol provides communications privacy for datagram protocols. It is based on the Transport Layer Security (TLS) protocol and offers equivalent security guarantees. Specifically, it ensures message confidentiality, integrity and authenticity, and supports protection against replay and Denial of Service attacks. DTLS reuses almost all protocol elements of TLS but introduces some important modifications in order to overcome the unreliability of the UDP transport protocol. It has, like TLS, a base protocol called *Record Layer*, and four sub protocols on top of it: the *Handshake*, the *ChangeCipherSpec* and the *Alert* protocol as well as the application data protocol.

The DTLS Record header format consists of the following fields: Content Type, indicating which sub protocol it is carrying, Protocol Version, Length, Epoch and Sequence Number. The last two fields were absent in the TLS record and were introduced to deal with lost records and records arrived out of order. The *Handshake* protocol is used to set up a new connection and negotiate the security parameters, like cipher suite, hash algorithms or compression. Unlike in TLS, the Handshake message includes explicit sequence numbering, the offset and the length of the fragment sent. Furthermore, to avoid the client and server deadlock due to message lost during the handshake, a retransmission mechanism has been introduced. In DTLS each end-point has a timer and keeps retransmitting previously sent message every time the timer expires until the next expected message is received.

III. IMPLEMENTATION ON EMBEDDED DEVICES AND TEST ENVIRONMENT

The performance comparison between the two security solutions has been conducted by using a test bed approach. This choice allowed us to analyse the real effectiveness of the implemented solutions as function of the hardware characteristics of the board used (e.g., clock speed, memory).

In the following, details about the implementation of both protocols on the Contiki Operating System are discussed, and, afterward, the selected hardware platforms and the test bed settings are described.

A. Implementation details

As explained above, AH and ESP can be used in tunnel mode or in transport mode. We selected transport mode only because the tunnel mode in the context of 6LoWPAN seems

not practical as additional headers would further increase the packet size. Furthermore, since we are interested in end-to-end security, the transport mode is the most suitable option. IPsec requires header compression to keep packet size reasonable in 6LoWPAN. We used LOWPAN_NHC encoding technique for AH and ESP proposed in [17] for the next header compression and LOWPAN_IPHC for IPv6 header compression described in [18]. We implemented AES-CTR for encryption in ESP and AES-XCBC-MAC-96 for authentication in AH and ESP. Moreover, we supplied a hardware implementation of AES algorithm as the application board used in this work comes with an AES 128 bit encryption accelerator. The length of Message Authentication Code consists of 96 bits. Protocols of automatic key exchange (e.g., IKE) are not supported in this implementation since asymmetric cryptography is assumed to be heavy weight for small sensor nodes. Pre-shared keys are used instead to establish SAs. The implementation of IPsec required the modification of the existing Contiki μ IP stack which already provides 6LoWPAN functionality and it is based on implementation of Shahid Raza [17].

DTLS was implemented as an application based on tinydtls library [19]. The cipher suite implemented is TLS_PSK_WITH_AES_128_CCM_8. The first acronym specifies that pre-shared mode is used to exchange the secret keys. We selected this mode for the same reason discussed for IPsec. CBC-MAC with Counter mode (CCM) provides both authentication and confidentiality and operates with AES algorithm. Also in this case we supplied a hardware implementation of AES algorithm. The length of Message Authentication Code is 8 bytes. We selected HMAC-SHA256 as keyed-hash message authentication code. This algorithm is used to generate DTLS cookies, required to prevent Denial of Service attacks, and to implement the PRF function (Pseudo Random Function). DTLS is not transparent to key exchange protocols and an implementation of handshake protocol is needed. However, the handshake process was simplified to reduce resources occupancy. The server message *Certificate*, *Server Key Exchange* and the client message *Certificate Verify* was completely removed since we did not employ asymmetric cryptography. We used LOWPAN_NHC encoding technique for UDP header compression and LOWPAN_IPHC for IPv6 header compression.

B. Hardware platforms

To compare the performance of the implemented security solutions two MB851 [20] rev. B (Figure 1), and one MB851 rev. D boards were used. They are developed by ST Microelectronics and are equipped with a 32-bit ARM® Cortex™-M3 microcontroller operating at a clock frequency up to 24 MHz. The MB851 rev. B boards embed 8 Kbytes of RAM and 128 Kbytes of eFlash as ROM, while the MB851 rev. D is equipped with 16 Kbytes of RAM and 256 Kbytes of eFlash. Moreover, all boards integrate a 2.4 GHz wireless transceiver compliant with the IEEE802.15.4 standard providing in hardware MAC functionality and a AES128 encryption accelerator.

The MB851 boards have been mainly selected due to the mounted microcontroller that is highly optimized to guarantee high performance at very low power consumption.



Fig. 1. The MB851 evaluation board.

C. Test bed settings and data collection scenario

The experimental campaigns have been carried out in an indoor environment. The evaluation setup is illustrated in Figure 2. It consists of two sensor nodes, a border router and a Linux machine running Debian OS. The laptop is connected to the border router through the Serial Line Internet Protocol (SLIP). Specifically, the end-system acts as UDP client, while the UDP server runs on one sensor node (implemented by the MB851 rev. D board). The other sensor node works as *sniffer* and is connected to a laptop running *Wireshark*. The border router is in charge of functionalities such as routing, compression, fragmentation and IPv6-802.15.4 translation, and is connected to a device equipped with Internet access.

The sensor node acting as server executes a simple UDP *echo* application, which listens to a fixed UDP port. When a packet is received, it is processed and, then sent back to the sender. Two different UDP clients for the Debian OS, one for each considered security solution, have been implemented.

In order to avoid the delay of a duty-cycled MAC layer, the Contiki's NULLRDC driver has been used in the experiments and hence the nodes kept their radio turned on all the time.

To evaluate the performance of the implemented security solutions, the following configurations have been compared: (i) IPsec with AH in transport mode, (ii) IPsec with ESP in

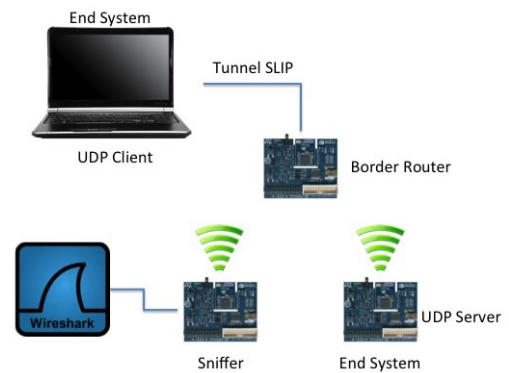


Fig. 2. Test bed scenario.

transport mode, (iii) DTLS, and (iv) an implementation without security. The analysis has been performed considering UDP datagrams with a payload ranging from 1 to 128 bytes. Several metrics have been evaluated: response time (i.e., the time the client takes to send a message, and to receive a response from the server), bandwidth usage (i.e., the amount of data transmitted and received by the client) and energy consumption (i.e., the energy consumed by the server node from receipt of the first fragment of a message, to the transmission of the last bit of the response message back to the client). In particular, the bandwidth usage, expressed in byte/s, has been measured considering a data rate of 5pkt/s by the client. During such analysis, also the response packets received from the server were considered. Furthermore, in the case of DTLS protocol even the handshake traffic has been used. The Energy Consumption is expressed in mJ and it has been determined by the following formula:

$$\text{Energy Consumption} = (\text{rxend} - \text{rxstart}) * I * V / \text{rticks} \quad .1$$

where $(\text{rxend}-\text{rxstart})$ is the number of system ticks (i.e., the system tick is the measure of time inside Contiki) elapsed between the two events previously mentioned; rticks is the number of system ticks per second (e.g., in Contiki 2.6 it is 117190); V represents the supply voltage; I is the current device consumption. In the specific case of our setup the value of current consumption used for MB851 device is reported in Table I.

The packet overhead and the memory footprint introduced by each considered implementation have been also evaluated. In particular, the overhead is expressed in bytes and it has been determined by the following formula:

$$\text{Packet Overhead} = \text{length}(802.15.4 \text{ MAC frame}) - \text{length}(\text{UDP payload}) \quad .2$$

It represents the sum of all headers' lengths added by the several protocol layers, including those of security. The Packet Overhead measures have been referred to the set up without security protocol with 6LoWPAN header compression mechanisms enabled. The memory footprint is obtained by *objdump* tool included within the Intel-ARM cross-tool chain.

Finally, let us observe that all tests have been carried out by using the independent replications method, and all results are characterized by a 95% confidence interval with a 5% maximum relative error.

IV. RESULTS

This section shows the performance of the analyzed

TABLE I. MB851 CURRENT CONSUMPTION

SoC state	I [mA]
CPU running mode	7,5
CPU sleep mode	3
Radio transmit	21
Radio receive	19

protocols in the 6LoWPAN scenario. The illustrated results have been obtained by the execution of real experimental tests implemented as described in the previous section.

Figure 3 shows the response time in dependency of the UDP data size. When the datagram size is too large to fit a single 802.15.4 packet, the data is fragmented according to the 6LoWPAN standard. In a similar way, 6LowPAN headers compression is activated whenever data size overcomes a prefixed compression threshold. We have to underline that response time for DTLS protocol doesn't include time spent for handshaking stage, because it occurs only once to establish a secure communication channel over a period of time. However we have evaluated it. Its average value is 158ms.

From the graph in Figure 3, we observe that IPSec AH performs better than IPSec ESP and DTLS. The explanation lies in the fact that AH does not provide encryption, but only authentication for IP header and payload. Moreover we also observe that ESP has a greater delay than DTLS because ESP, being a network layer protocol, encrypts and authenticates also the transport header. DTLS on the contrary does not protect in any way the UDP header, but only authenticates and encrypts its payload. In the worst case scenario, AH has a response time overhead of 10ms referred to the response time measured without security protocols, while for ESP and DTLS the overhead is around 14ms.

From the graph in Figure 4 we observe that, for UDP data size of few bytes, the case without security protocols performs worse than IPsec AH. This behavior is justified by the lack of 6LoWPAN headers compression under a specific network packet length. The data overhead of security protocols, instead, triggers the compression mechanism independently of the payload size. Obviously when payload size increases, and header compression is instigated, the case without security solution performs better than all the others.

DTLS is the protocol that requires more bandwidth in every situation and this is due to the handshake process not present in IPsec. The trends of DTLS and ESP are essentially the same. In fact, as we will see later, the packet overhead introduced by both protocols is equal. The factor that positions the curve DTLS above ESP in the graph of Figure 4 is the

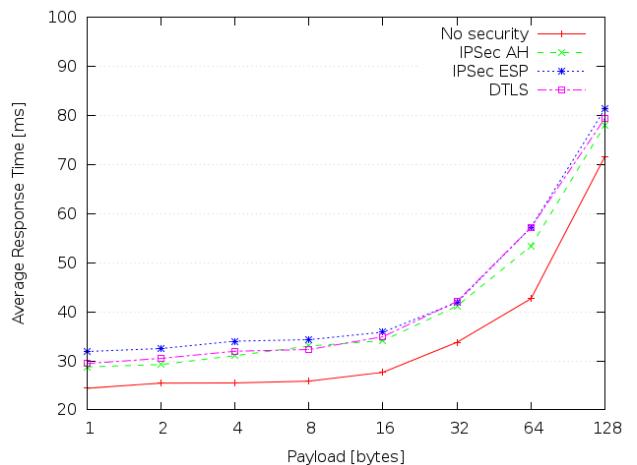


Fig. 3. Response time in dependency of the UDP data size.

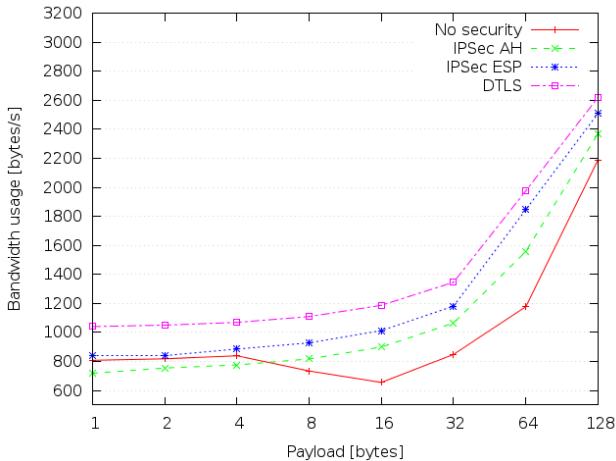


Fig. 4. Bandwidth usage in dependency of the UDP data size.

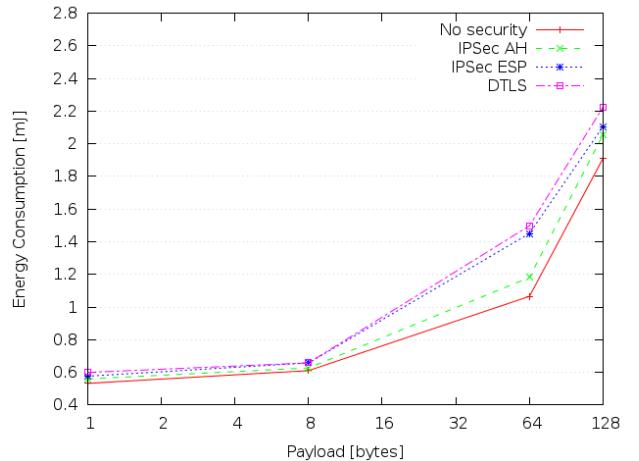


Fig. 5. Energy consumption in dependency of the UDP data size.

presence of the negotiation mechanism that instead is missing in ESP. In the worst case scenario, AH produces an increase of used bandwidth of 32% compared to the solution without security, while ESP and DTLS involve an increase of used bandwidth respectively 58% and 67%.

The graph in Figure 5 shows energy consumption versus UDP payload. For small payload we observe that the analyzed security protocols are very close to one another because major consumption is due to the radio transmit and receive, and the CPU is loaded in the same way. DTLS affected by the presence of the handshake phase, has a power consumption greater than AH and ESP for each payload size. AH instead turns out to be the more economic security solution in terms of energy and this is justified by the fact that the CPU is used only for calculating the MAC while in the other solutions the data is also encrypted.

By increasing the payload the differences in energy consumption become more significant because the CPU is busy performing cryptographic operations on a bigger amount of data. The worst-case scenario is represented by DTLS, which consumes up to 40% more than the application without security. AH and ESP in the worst cases increase the power consumption by up to 11% and 36% respectively.

The histogram in Figure 6 represents the packet overhead introduced by each of the analyzed protocol. The overhead has been calculated as the difference between the length of the IEEE 802.15.4 frame and the UDP payload and therefore it represents the sum of all the headers added by the various layers of protocol stack (i.e., including the security layers). Note that, the overhead computation was carried out while maintaining the 6LoWPAN header compression enabled.

The histogram in Figure 7 reports the measures of RAM and ROM footprints of IPsec AH, IPsec ESP and DTLS implementation. Although the analyzed protocols share the same cryptosystem, DTLS implementation results more complex than AH and ESP. In fact DTLS includes handshake protocol, management of retransmission timers, keys and cookies generation, that are not present into AH and ESP implementation.

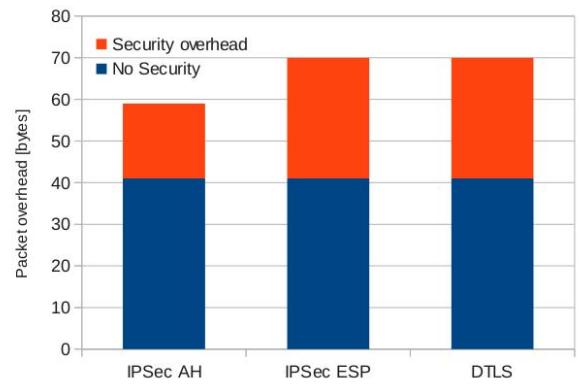


Fig. 6. Packet overhead introduced by each of the analyzed protocol.

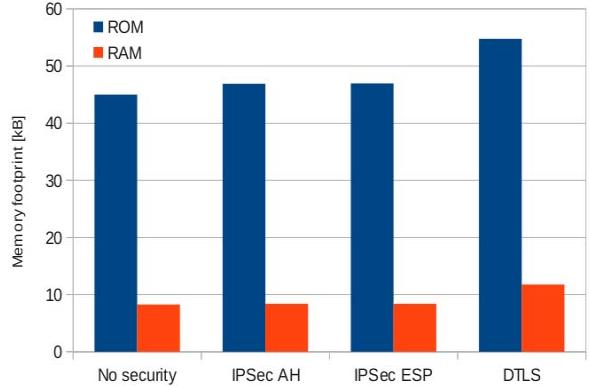


Fig. 7. RAM and ROM footprints of IPsec AH, IPsec ESP and DTLS implementation.

V. CONCLUSIONS

The widespread diffusion of constrained networks, as a part of the Internet of Things, will gradually make the security

aspect more and more important in order to ensure advanced services to end users. Security protocols currently used on the public Internet work well on traditional end-systems, but they require excessive resources, and therefore they are not adequate for constrained devices. In this paper, we have adapted the existing implementations of IPSec and DTLS security protocols for WSNs to make them work fine on our IoT devices. Furthermore, we have implemented specific applications to evaluate the performance of both protocols and their impact on the limited resources of embedded devices. The results showed that both implementations are able to ensure an adequate level of end-to-end security within a WSN. However, it was not possible to establish unequivocally an optimal solution for any applicative scenario. The choice of one of the two protocols is closely related to the requirements of the particular application in which the embedded devices are used.

REFERENCES

- [1] L. Mainetti, L. Patrono, and A. Vilei, "Evolution of wireless sensor networks towards the Internet of Things: A survey". In Proc. of the 19th International Conference on Software, Telecommunications and Computer Networks (SoftCOM 2011). Split, Croatia, 2011, pp. 1 – 6.
- [2] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks". RFC4944, 2007.
- [3] Z. Shelby, K. Hartke, C. Bormann, and B. Frank, "Constrained Application Protocol (CoAP)". draft-ietf-core-coap-06, 2011.
- [4] D. Alessandrelli, L. Mainetti, L. Patrono, G. Pellerano, M. Petracca, M.L. Stefanizzi, "Implementation and validation of an energy-efficient MAC scheduler for WSNs by a test bed approach". In Proc. of the 20th International Conference on Software, Telecommunications and Computer Networks (SoftCOM 2012). Split, Croatia, 2012.
- [5] D. Alessandrelli, L. Mainetti, L. Patrono, G. Pellerano, M. Petracca, M.L. Stefanizzi, "Performance Evaluation of an Energy-Efficient MAC Scheduler by using a Test Bed Approach", Journal of Communication Software and Systems, vol 9, no 1, 2013.
- [6] L. Catarinucci, S. Guglielmi, L. Patrono, and L. Tarricone, "Switched-beam antenna for wireless sensor network nodes," Progress In Electromagnetics Research C, V ol. 39, 193-207, 2013.
- [7] L. Catarinucci, S. Guglielmi, L. Mainetti, V. Migali, L. Patrono, M.L. Stefanizzi and L. Tarricone, "An Energy-Efficient MAC Scheduler based on a Switched-Beam Antenna for Wireless Sensor Networks", Journal of Communication Software and Systems, Vol.9 No.2, 2013, pp. 117 – 127.
- [8] J. Granjal, E. Monteiro, and J. Sa Silva, "Enabling Network-Layer Security on IPv6 Wireless Sensor Networks". IEEE Global Telecommunications Conference (GLOBECOM 2010), Miami, FL, Dec. 2010, pp. 1 – 6.
- [9] S. Raza, D. Trabalza, and T. Voigt, "6LoWPAN Compressed DTLS for CoAP". IEEE 8th International Conference on Distributed Computing in Sensor Systems (DCOSS), Hangzhou, May 2012, pp. 287 – 289.
- [10] A. Sehgal, V. Perelman, S. Kuryla, J. Schonwalder, "Management of resource constrained devices in the internet of things". IEEE Communications Magazine, Vol. 50 No. 12, Dec. 2012, pp. 144 – 149.
- [11] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki – a lightweight and flexible operating system for tiny networked sensors". First IEEE Workshop on Embedded Networked Sensors, Tampa, Florida, USA, November 2004.
- [12] S. Kent and K. Seo: "Security Architecture for the Internet Protocol". RFC 4301, December 2005.
- [13] S. Kent: "IP Authentication Header". RFC 4302, December 2005.
- [14] S. Kent: "IP Encapsulating Security Payload (ESP)". RFC 4303, December 2005.
- [15] C. Kaufman, P. Hoffman, Y. Nir, and P. Eronen: "Internet Key Exchange Protocol Version 2 (IKEv2)". RFC 5996 (Proposed Standard), September 2010.
- [16] E. Rescorla: "Datagram Transport Layer Security Version 1.2". RFC 6347, January 2012.
- [17] S. Raza, S. Duquennoy, T. Chung, D. Yazar, T. Voigt, U. Roedig, "Securing communication in 6LoWPAN with compressed IPsec", Proc. of 2011 International Conference on Distributed Computing in Sensor Systems and Workshops, DCOSS'11, Barcelona; Spain; 27 June 2011.
- [18] J. Hu: "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks". RFC 6282, September 2011.
- [19] tinydtls Library, <http://tinydtls.sourceforge.net> (Accessed May 2013).
- [20] MB851 user manual, http://www.st.com/internet/com/TECHNICAL_RESOURCES/TECHNICAL_LITERATURE/USER_MANUAL/CD00262415.pdf (Accessed May 2013).