On experimentally evaluating the impact of security on IEEE 802.15.4 networks

Roberta Daidone, Gianluca Dini and Marco Tiloca
Department of Ingegneria dell'Informazione, University of Pisa
Largo Lazzarino 1, 56100 Pisa, Italy
email: {r.daidone, g.dini, m.tiloca}@iet.unipi.it

***Abstract*—IEEE 802.15.4 addresses low-rate wireless personal area networks, enables low power devices, and includes a number of security provisions and options (the security sublayer). Security competes with performance for the scarce resources of low power, low cost sensor devices. So, a proper design of efficient and secure applications requires to know the impact that IEEE 802.15.4 security services have on the protocol performance. In this paper we present the preliminary results of a research activity aimed at quantitatively evaluating such impact from different standpoints including memory consumption, network performance, and energy consumption. The evaluation exploits a free implementation of the IEEE 802.15.4 security sublayer.**

***Keywords*-IEEE 802.15.4; security; performance evaluation;**

I. Introduction

IEEE 802.15.4 is an emerging standard addressing the needs of low-rate wireless personal area networks with a focus on enabling low power devices, personal area networks, and wireless sensor networks (WSNs). The standard is characterized by maintaining a high level of simplicity, allowing for low cost and low power implementations [1].

IEEE 802.15.4 is adopted in a wide range of application scenarios including environmental monitoring, health-care, military surveillance, and industrial automation. Most of these applications require forms of secure communication including confidentiality, authenticity, and ready detection of replay-attacks. For this reason, IEEE 802.15.4 specification includes a number of security provisions and options.

Security and performance of IEEE 802.15.4 have been extensively analysed, although separately. Relevant works include [2], [3] as to security analysis, and [4], [5] as to performance analysis. In contrast, a thorough analysis of the impact that security provisions and options have on IEEE 802.15.4 performance is missing. Some related works have been presented, but they face either with specific aspects, such as cipher design [6], or with collateral although important issues, such as key management [7], [8]. What it is really missing is an analysis providing quantitative indications regarding the consumption of system resources due to security. We believe that this analysis is crucial. Security and performance compete for the same system resources, namely memory, CPU, bandwidth and energy, that are scarce in low power, low cost sensor devices. Therefore, quantitative indications regarding resources consuption are fundamental to design and implement adequate performance-security trade-offs in IEEE 802.15.4-based applications.

For these reasons we have recently started a research activity aimed at experimentally evaluating impact and costs of IEEE 802.15.4 security services. In this activity we refer to a free implementation of the IEEE 802.15.4 specification for TinyOS on TmoteSky motes [9]. We have extended it with an implementation of the IEEE 802.15.4 security sublayer, which is compliant to the standard specification [10]. To the best of our knowledge, this is the first available free implementation of IEEE 802.15.4 including security services.

Our experimental evaluation focuses on three main performance aspects, namely memory occupancy, network performance, and energy consumption, and has a twofold objective. On the one hand, we aim at evaluating how security impacts on these three aspects. In particular, we are interested in determining how security services (e.g. confidentiality and/or authenticity) and security options (e.g. message integrity code length) influence such aspects. On the other hand, we are willing to devise a model that allows designers and implementers to carry out, for example at re-deployment, simulative and/or analytical performance analysis that include security too.

In this paper we report some preliminary results of our activity and presents some future steps and goals. As to *memory occupancy*, we show that security requires a non negligible although affordable amount of memory. This result is relevant as WSNs often comprise devices whose storage capabilities are severely limited. For instance, TmoteSky motes have 48 Kbytes of available memory and our implementation of the IEEE 802.15.4 security sublayer requires just the 9.4% of that memory on the sender side, and the 12.9% on the receiver side.

As to *network performance*, we show that the security impact derives from the fact that, in the most general case, secured frames are larger than unsecured ones (frame expansion) and that securing frames requires additional frame processing (extra processing). We show that securing communications reduces the amount of transmitted data frames of up to 33.8%.

Finally, frame expansion and extra processing influence *energy consumption*, which is one of the main issues in WSNs. We show that the major impact is due to the

transmission of expanded data frames, which represents the 61.12% of the overall extra energy consumption in the presence of security.

The remainder of this paper is organized as follows. Section II provides an overview of the IEEE 802.15.4 security services. In Section III, we briefly describe our implementation of the IEEE 802.15.4 security sublayer. Section IV reports preliminary results concerning memory consumption, network performance, and energy consumption. Section V presents our future works. Finally, in Section VI we draw our conclusive remarks.

II. On security in IEEE 802.15.4

Two different kinds of device can participate in an IEEE 802.15.4 network: *Full-Function Devices* (*FFDs*) and *Reduced-Function Devices* (*RFDs*). In particular, one FFD is elected as the *Personal Area Network* (*PAN*) Coordinator and is responsible for network and security management.

In the remainder of this paper, we consider a *beacon enabled PAN*, that is the PAN Coordinator periodically broadcasts *beacon* frames within the network. Specifically, the MAC attribute *BeaconOrder* defines the interval at which the PAN Coordinator broadcasts beacon frames.

IEEE 802.15.4 provides a number of security services and makes them available to the higher layers. In particular, data confidentiality, data authenticity and replay protection are supported on a per-frame basis. The standard includes a security suite based on the *Advanced Encryption Standard* (*AES*) 128 bits symmetric-key cryptography. The security suite relies on three elements: an *Auxiliary Security Header* (*ASH*), *security modes* and settings, and *security procedures*.

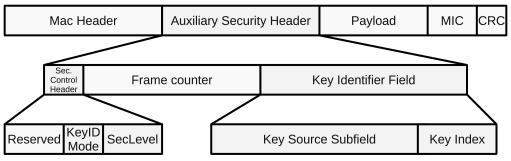

Figure 1. The Auxiliary Security Header (ASH).

If communications are secured, senders build the ASH, insert it next to the standard MAC header (see Figure 1), and secure frames before transmitting them. According to the information carried within the ASH, recipients retrieve the right cryptographic key and correctly unsecure MAC frames. More in details, the ASH carries information required for the security processing, including i) the specified *security mode* and its options, ii) a frame counter value for the anti-replay service, and, finally, iii) the *KeyIdMode* according to which the key retrieval procedure is supposed to take place.

In particular, the KeyIdMode indicates whether the cryptographic key has to be obtained implicitly or explicitly. The IEEE 802.15.4 standard provides four different KeyIdModes, that is four different ways to retrieve cryptographic keys. More specifically, in *KeyIdMode0* the key is determined implicitly from the originator and the recipient(s) of the frame, and the Key Identifier field of the ASH is not present. In *KeyIdMode1* the key is determined from the Index subfield of the Key Identifier field of the ASH, in conjunction with the macDefaultKeySource value, which is predetermined by network devices. In *KeyIdMode2* the key is determined explicitly from the 4 bytes Key Source subfield and the 1 byte Key Index subfield of the Key Identifier field. Finally, in *KeyIdMode3* the key is determined in the same way as in KeyIdMode2, but the Key Source subfield is 8 bytes in size instead of 4.

Security mode	**Data confidentiality**	**Data authenticity**	**MIC size (bytes)**
CTR	ON	OFF	-
CBC_MAC_4	OFF	ON	4
CBC_MAC_8	OFF	ON	8
CBC_MAC_16	OFF	ON	16
CCM_4	ON	ON	4
CCM_8	ON	ON	8
CCM_16	ON	ON	16

Table I
SECURITY MODES.

Three different kinds of security modes are available: encryption only (*CTR*), authentication only (*CBC_MAC*), and both encryption and authentication (*CCM*). In particular, CBC_MAC and CCM rely on a *Message Integrity Code* (*MIC*), whose size can be either 4, 8, or 16 bytes. By choosing and properly setting the security mode to be used, it is possible to deal with applications' constraints and security requirements. Table I provides an overview of the available security modes.

By means of the standard *security procedures*, it is possible to secure and unsecure MAC frames, assure a minimum security level, retrieve cryptographic keys, deal with blacklisted nodes, and verify frames' freshness by means of the *Frame Counter* field of the ASH. Securing/unsecuring operations rely on a fresh *nonce* value, that is a randomly generated number used to prevent replay attacks. Nonces are generated by senders and checked by recipients.

Security material is stored into two different tables on each device, that is a *Key Table* and a *Device Table*. The former contains cryptographic keys and their identifiers, while the latter includes information about other sender devices, such as the highest frame counter value received by each one of them. Security procedures are responsible for accessing and updating data structures, as well as verifying their consistency.

Finally, IEEE 802.15.4 does not concern about key establishment and devices authentication, which are potentially entrusted to the higher layers. Thus, both senders and recipients have to share common security settings and store the necessary security material before secure communications can actually take place.

III. Implementation of the security sublayer

We extended the open source implementation of IEEE 802.15.4 currently available for the TinyOS platform at [9]. In particular, we implemented IEEE 802.15.4 security services and procedures responsible for protecting MAC data frames, with reference to the TmoteSky motes [11] and the CC2420 chipset [12]. The source code of our implementation can be found at [13].

MAC layer security relies on two main sets of services, namely frame handling and actual security procedures. *Frame handling* has been extended in order to properly manage the auxiliary security header in case data frames require to be protected. *Security procedures* have been implemented on both the sender and the receiver side, according to the guidelines and practices described by the IEEE 802.15.4 standard.

While implementing our security suite, we made reference to the security mechanisms provided by the CC2420 chipset. All security modes described in Section II are available and can be selected on a per-frame basis, according to the application requirements on the sender side. CC2420 provides cryptographic primitives based on AES 128 bits encryption, and hardware support for the IEEE 802.15.4 security services. MAC frames protection can be performed in two different ways: *stand-alone* or *in-line*. The former encrypts MAC frames into a proper RAM buffer, while the latter secures and unsecures frames within the transmit buffer TXFIFO and the receive buffer RXFIFO, respectively. In the rest of this paper, we refer to the in-line security operations.

CC2420 determines the security mode to be used according to the SECCTRL0 and SECCTRL1 registers settings. That is, these registers have to be properly set before the actual security operations take place. Besides, before issuing a security command strobe, it is necessary to set the cryptographic key to be used into the KEY0 or KEY1 register. Finally, in order to detect replay attacks, a nonce is written in the TXNONCE (on the sender side) or RXNONCE (on the recipient side) register.

Outgoing frames protection is accomplished by issuing the STXENC command strobe, which actually secures the frame within the TXFIFO buffer and then transmits it. On the other hand, recipient nodes invoke the SRXDEC command strobe, which unsecures the frame inside RXFIFO and makes it available to the higher layers.

IV. Performance evaluation

In order to test our security sublayer, we have used two TmoteSky motes, equipped with a 48 Kbytes ROM, and a 8 MHz MSP430 microcontroller with a 10 Kbytes RAM. In particular, we have considered a beacon enabled network with BeaconOrder 7. An FFD acts as the PAN Coordinator and a single RFD acts as sender. Both the RFD and the FFD share a common cryptographic symmetric key. The PAN Coordinator unsecures received protected data frames, and sends ACK frames back. On the other hand, the RFD continuously transmits protected data frames to the PAN Coordinator, and waits for ACK frames back. We consider data frames whose payload is 18 bytes in size. Since the IEEE 802.15.4 standard does not permit to secure ACK frames, they are neither encrypted nor authenticated.

A. Memory consumption

The amount of available memory on a TmoteSky mote may represent a severe constraint while developing applications or, as in our case, while extending MAC layer capabilities. We have evaluated the memory overhead due to the presence of IEEE 802.15.4 security sublayer by comparing the amount of used memory both in the presence and in the absence of security. In the absence of security, memory is necessary to allocate the application and TinyOS images. In the presence of security, additional memory is necessary to allocate security services and data structures. As to security, we have considered the KeyIdMode2 and the CCM_16 security mode.

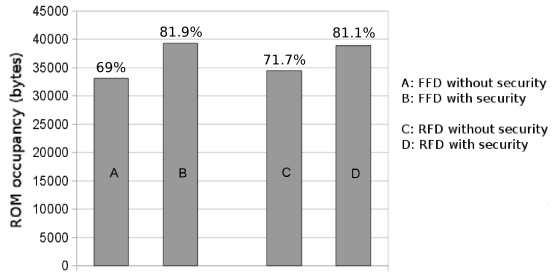

Figure 2. Memory consumption.

Figure 2 shows the memory footprints of the PAN Coordinator (columns A, B) and the RFD (columns C, D) with and without security services, respectively. Y-axis reports the absolute memory occupancy in bytes, whereas percentages express memory occupancy as a fraction of the available memory (i.e. 48 Kbytes).

On the PAN Coordinator, memory occupancy without security is 33.12 Kbytes (i.e. 69% of the available memory). It becomes 39.31 Kbytes (81.9%) when security is used. It follows that security causes an increase of memory occupancy of about 6.19 Kbytes (12.9%). This leaves 8.69 Kbytes (18.1%) free for other uses. As to the RFD, memory occupancy without security is 34.43 Kbytes (71.7%), whereas it becomes 38.94 Kbytes (81.1%) with security. It follows that security causes an increment of 4.51 Kbytes (9.4%). This leaves 9.06 Kbytes (18.9%) free for other uses. Without security, the difference in size between the RFD and the PAN Coordinator is merely due to their different basic operations. In the presence of security, the memory occupancy increases both on the RFD and the PAN Coordinator. However, unlike RFDs, the PAN Coordinator

is required to deal with larger security structures in a more complex way.

Notice that the amount of free memory is important because other high level security services might be necessary. For instance, let us consider key establishment. If a designer wishes to use Elliptic Curve Diffie-Hellman (ECDH), a possibility is the implementation in the TinyECC suite [14]. However, only the unoptimized version could be accommodated in the free memory as it requires 4446 bytes on a TmoteSky mote. Of course, a more memory-efficient implementation of IEEE 802.15.4 security sublayer could save more memory. However, although we believe that our implementation is affordable given the limited memory overhead it causes, we claim that our aim is not to achieve a highly optimized implementation but, rather, to provide an early experimental framework to start quantitative evaluations.

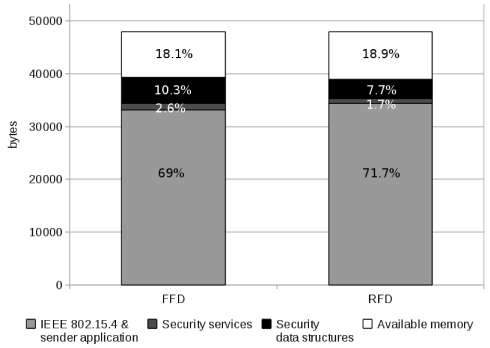

Figure 3. Memory occupancy breakdown.

Furthermore, we have considered the impact of data structures and security operations, separately. Figure 3 shows the memory usage breakdowns on the PAN Coordinator and the RFD side, respectively. In particular, the PAN Coordinator requires 1228 bytes (2.6%) for security services and 4958 bytes (10.3%) for security data structures and their relative operations. The RFD requires 826 bytes (1.7%) for security services and 3688 bytes (7.7%) for security data structures and their relative operations.

It is important to notice that memory consumption of data structures varies with the number of RFDs and cryptographic keys, whereas memory consumption of the code does not. Intuitively, the Device Table size grows with the number of RFDs, whereas the Key Table grows with the number of keys. However, the number of RFDs and the number of keys may depend on the design choices. For instance, one possible choice is that all devices in the network share a single "network key". An alternative choice is that each RFD may share a private secret key with the PAN Coordinator. A thorough evaluation of memory consumption of data structures with respect to the number of RFDs and cryptographic keys is thus one of the next steps.

B. Impact of security on network performance

In this section we discuss the impact of security on network performance by evaluating the decrement of transmitted data frames due to security. Operatively, we consider a single RFD continuously transmitting data frames to the PAN Coordinator for a given amount of time T, as described at the beginning of Section IV. Then, we count the number of both secured and unsecured frames transmitted during such an amount of time. Finally, we perform the subtraction of the latter number of frames from the former. As to secured communication, we consider key retrieval mode KeyIdMode2 and all security modes. Provided results are averaged over ten repetitions lasting $T = 120$ s each. In order to count data frames without introducing any additional delay, we used a Texas Instruments IEEE 802.15.4/Zigbee packet sniffer equipped with a CC2430 chipset [15].

Mode	Parameter	# frames	Decrement (%)	Frame size (bytes)
No security	F	7685.5	-	27
CTR	F_e	5671.6	26.2	37
CBC_MAC_4	F_{a_4}	5534.8	28	41
CBC_MAC_8	F_{a_8}	5371.6	30.1	45
CBC_MAC_16	$F_{a_{16}}$	5110.3	33.5	53
CCM_4	F_{ea_4}	5514.6	28.2	41
CCM_8	F_{ea_8}	5374.4	30.1	45
CCM_16	$F_{ea_{16}}$	5082.1	33.9	53

Table II
NUMBER OF TRANSMITTED FRAMES VS. SECURITY MODES.

Table II shows the number of transmitted data frames in the different security modes. The column "Mode" lists security modes, and "No security" specifies that frames are not secured. The parameter F refers to the number of data frames transmitted when security is not used, whereas F_e, $F_{a_x}(x = 4, 8, 16)$ and $F_{ea_x}(x = 4, 8, 16)$ specify the number of data frames transmitted when encryption (e), authentication (a) or both (ea) are used, respectively. When authentication is used, x specifies the MIC size. The column "Frame size" specifies the size of the frame for each security mode. Finally, the column "Decrement" specifies the transmitted data frame decrement for each security mode, with respect to the "No security" mode. As it turns out, security causes quite a tangible transmitted data frame decrement, ranging from 26.2% in the CTR mode, up to 33.9% in the CCM_16 mode.

The overhead introduced by security services consists in two elements, the *communication overhead* C and the *processing overhead* P. The communication overhead C is due to the extra bytes that it is necessary to transmit in the presence of security. These extra bytes account for the additional ASH and the MIC field. The processing overhead P is due to the extra processing necessary to parse the ASH, compute the MIC, and secure data frames. The objective of our analysis is to separate the contributions

of communication and processing from the total overhead. We have accomplished such a task in the case of CCM_16, which causes the largest overhead (see Table II) from both the processing (encyphering and hashing) and the communication (largest MIC) viewpoint.

The communication overhead C has been evaluated as the difference between F, the number of transmitted data frames in the "No security" mode, and $F^*_{a_{16}}$, the number of data frames transmitted when frames have the same size as frames in CBC_MAC_16 mode (see Table II). That is, $C = (F - F^*_{a_{16}})$. In order to evaluate $F^*_{a_{16}}$, we have transmitted unsecured data frames whose payload (18 bytes) has been increased by the size of the MIC field (16 bytes) and the ASH (10 bytes), for a total of 44 bytes. $F^*_{a_{16}}$ amounts to 6091.7 data frames. Note that frames are pre-prepared in order to avoid any processing overhead. The communication overhead C results in a decrement of transmitted frames equal to 1593.8. As CCM_16 causes a total decrement equal to $(7685.5 - 5082.1) = 2603.4$ frames (see Table II), then the communication overhead represents the 61.2% of the overall overhead (i.e. 2603.4 data frames).

The processing overhead P can be now determined as the difference between the total overhead and the communication overhead, i.e. $P = (F - F_{ea_{16}} - C)$. P results equal to 1009.6, unsent data frames, which amounts to the 38.8% of the whole overhead. Note that P is given by three processing overhead subcomponents, namely i) building and handling the ASH, ii) computing the MIC, and, finally, iii) encrypting the frame. The fine-grain evaluation of these components will be the goal of the next future work.

C. Impact of security on energy consumption

While evaluating how security impacts on WSNs, it is also important to consider the costs in terms of energy consumption due to the additional communications and computations required by any given security mode. More in details, a security mode requires the following operations: i) extra communications for the transmission of the ASH and the MIC; ii) extra computation at the hardware level for encryption/decryption and/or generation/verification of the MIC; iii) extra computation for security management, e.g. retrieving keys from the Key Table or ASH parsing. Operations i) and ii) are performed by CC2420, while operation iii) involves the MSP430 microcontroller. As a consequence, the extra energy consumption E of a given security mode is given by $E = E_c + E_s + E_p$, where E_c, E_s, and E_p are the energy consumptions of i), ii), and iii), respectively.

Each component $E_x, x \in \{c, s, p\}$, is evaluated as $E_x = V_x \times I_x \times t_x$, where V_x and I_x are, respectively, the supply voltage and the absorbed current of the hardware device performing operation x, and t_x is the duration of the operation. The time interval t_x has been computed according to the same experimental method described in Section IV-B, by properly bypassing operations not involved in the energy component under exam. The values of V_x and I_x are those specified by the CC2420 and TmoteSky motes data sheets [12][11]. If security is on, we refer to the KeyIdMode2 and the CCM_16 security mode, that is the security mode which causes the largest overhead.

E	V	I	t	Involved component
$E_c = 240.54\ \mu J$	$3.6\ V$	$17.4\ mA$	$3.84\ ms$	CC2420
$E_s = 150.34\ \mu J$	$3.6\ V$	$17.4\ mA$	$2.40\ ms$	CC2420
$E_p = 2.66\ \mu J$	$3\ V$	$600\ \mu A$	$1.48\ ms$	MSP430

Table III
ENERGY CONSUMPTION OVERVIEW.

Table III provides the energy consumption components, and reports related supply voltage, current consumption, and time intervals. More details follow.

- $E_c = V_c \times I_c \times t_c$ is the additional energy comsumed to transmit one secured data frame because of the presence of the ASH and the MIC. Since frames transmission involves the CC2420 chipset, the supply voltage $V_c = 3.6\ V$ and the current consumption $I_c = 17.4\ mA$ have been considered. The time t_c has been calculated as $t_c = t_{c_{ea_{16}}} - t_{c_0}$. In particular, $t_{c_{ea_{16}}}$ is the time required to transmit an unsecured data frame with a 44 bytes payload, thus simulating the presence of the ASH and the MIC. On the other hand, t_{c_0} is the time required to transmit an unsecured basic data frame with a 18 bytes payload.
- $E_s = V_s \times I_s \times t_s$ is the additional energy comsumed to perform hardware encryption and authentication. Since these operations involve the CC2420 chipset, the supply voltage $V_s = 3.6\ V$ and the current consumption $I_s = 17.4\ mA$ have been considered. The time t_s has been calculated as $t_s = t_{s_{ea_{16}}} - t_{s_0}$. In particular, $t_{s_{ea_{16}}}$ is the time required to transmit a secured data frame having a 18 bytes payload, performing all hardware security operations but not the security management operations. On the other hand, t_{s_0} is the time required to transmit an unsecured basic data frame with a 18 bytes payload.
- $E_p = V_p \times I_p \times t_p$ is the additional energy comsumed to perform security management operations. Since these operations involve the MSP430 microcontroller, the supply voltage $V_p = 3\ V$ and the current consumption $I_p = 600\ \mu A$ have been considered. The time t_p has been calculated as $t_p = t_{p_{ea_{16}}} - t_{p_0}$. In particular, $t_{p_{ea_{16}}}$ is the time required to transmit a data frame having a 18 bytes payload, performing all security management operations but not the actual hardware security operations. On the other hand, t_{p_0} is the time required to transmit an unsecured data frame with a 44 bytes payload, thus simulating the presence of the ASH and the MIC.

Finally, the overall extra energy consumption due to security is $E = E_c + E_s + E_p = 393.54\ \mu J$ per data frame.

V. Future works

In this paper, we have presented some preliminary experimental results. Our future research activities aim at investigating IEEE 802.15.4 security impact more in details. Again, we will take into account how security affects memory occupancy, network performance, and energy consumption.

As to memory occupancy, we will thoroughly evaluate memory consumption of data structures when varying the number of RFDs and cryptographic keys. Then, we will make a fine-grain evaluation of the transmission and processing overhead due to the presence of security services. In particular, we will investigate specific contributions related to security operations, such as encryption and authentication. Also, we will discuss more precisely how different security modes and KeyIdModes contribute to the overall energy consumption of TmoteSky motes. Finally, we will consider the above mentioned issues while varying the data frames payload in size.

VI. Conclusion

In this paper, we have presented our implementation of the IEEE 802.15.4 security sublayer for the TinyOS platform and the TmoteSky motes. To the best of our knowledge, there are no other free available implementations of the IEEE 802.15.4 security sublayer for this architecture to date.

Then, we have described how we used our implementation to experimentally evaluate security impact and costs of IEEE 802.15.4 security services. In particular, we have focused on memory occupancy, network performance, and energy consumption, and presented some preliminary results.

We have shown that IEEE 802.15.4 security services require an affordable amount of memory, and have a meaningful impact on network performance and energy consumption. We believe that this work is a first step towards a quantitative analysis that allows designers and implementers to properly define a security-performance trade-off.

Acknowledgments

This work has been supported by EU FP7 Network of Excellence CONET (Grant Agreement no. FP7-224053), EU FP7 Project CHAT (Grant Agreement no. FP7-224428), and EU FP7 Integrated Project PLANET (Grant agreement no. FP7-257649).

References

[1] *IEEE Std. 802.15.4-2006, IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)*, Institute of Electrical and Electronics Engineers, Inc., New York, September 2006.

[2] N. Sastry and D. Wagner, "Security considerations for IEEE 802.15.4 networks," in *Proceedings of the 3rd ACM workshop on Wireless security*, ser. WiSe '04. New York, NY, USA: ACM, 2004, pp. 32–42.

[3] V. B. Mišić, J. Fang and J. Mišić, "MAC layer security of 802.15.4-compliant networks," *IEEE International Conference on Mobile Adhoc and Sensor Systems Conference*, vol. 0, p. 854, 2005.

[4] J. Zheng, M. J. Lee, and M. Anshel, "Toward secure low rate wireless personal area networks," *IEEE Trans. Mob. Comput*, vol. 5, no. 10, pp. 1361–1373, 2006.

[5] J.-S. Lee, "Performance evaluation of ieee 802.15.4 for low-rate wireless personal area networks," *IEEE Transactions on Consumer Electronics*, vol. 52, no. 3, pp. 742–749, August 2006.

[6] Y. Xiao, H. H. Chen, B. Sun, R. Wang and S. Sethi, "MAC security and security overhead analysis in the IEEE 802.15.4 wireless sensor networks," *EURASIP J. Wirel. Commun. Netw.*, vol. 2006, pp. 81–81, April 2006.

[7] F. Amini, M. Khan, J. Mišić and H. Pourreza, "Performance of IEEE 802.15.4 Clusters with Power Management and Key Exchange," *Journal of Computer Science and Technology*, vol. 23, pp. 377–388, 2008.

[8] J. Mišić, "Cost of secure sensing in IEEE 802.15.4 networks," *IEEE Transactions on Wireless Communications*, vol. 8, no. 5, pp. 2494–2504, 2009.

[9] "Source code of the IEEE 802.15.4 MAC Implementation." [Online]. Available: http://code.google.com/p/tinyos-main/source/browse/trunk/tos/lib/mac/tkn154/

[10] J.-H. Hauer, R. Daidone, R. Severino, J. Büsch, M. Tiloca and S. Tennina, "Poster Abstract: An Open-Source IEEE 802.15.4 MAC Implementation for TinyOS 2.1," in *Proceedings of 8th European Conference on Wireless Sensor Networks (EWSN)*, February 2011.

[11] "Tmote iv Low Power Wireless Sensor Module." [Online]. Available: http://www.bandwavetech.com/download/tmote-sky-datasheet.pdf

[12] "Texas Instruments CC2420 2.4 GHz IEEE 802.15.4 / ZigBee ready RF Transceiver." [Online]. Available: http://focus.ti.com/lit/ds/symlink/cc2420.pdf

[13] "Source code of the IEEE 802.15.4 security sublayer implementation." [Online]. Available: http://www.iet.unipi.it/g.dini/download/Tinyos_security.zip

[14] A. Liu and P. Ning, "TinyECC: A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks," in *Proceedings of the 7th International Conference on Information Processing in Sensor Networks (IPSN 2008)*, St. Louis, Missouri, USA, April 22-24 2008, pp. 245–256.

[15] "Texas Instruments CC2430DB." [Online]. Available: http://focus.ti.com/docs/toolsw/folders/print/cc2430db.html