Journal of Universal Computer Science, vol. 19, no. 17 (2013), 2490-2502
submitted: 15/2/13, accepted: 22/4/13, appeared: 1/11/13 © *J.UCS*

Enabling User Access Control in Energy-constrained Wireless Smart Environments

Juan Álvaro Muñoz Naranjo
(Dpt. of Computer Science, University of Almería
Agrifood Campus of International Excellence (ceiA3), Spain
jmn843@ual.es)

Pablo Orduña
(Deusto Institute of Technology - DeustoTech, University of Deusto
Bilbao, Spain
pablo.orduna@deusto.es)

Aitor Gómez-Goiri
(Deusto Institute of Technology - DeustoTech, University of Deusto
Bilbao, Spain
aitor.gomez@deusto.es)

Diego López-de-Ipiña
(Deusto Institute of Technology - DeustoTech, University of Deusto
Bilbao, Spain
dipina@deusto.es)

Leocadio González Casado
(Dpt. of Computer Science, University of Almería
Agrifood Campus of International Excellence (ceiA3), Spain
leo@ual.es)

Abstract: This work introduces a novel access control solution for wireless network services in Internet of Things scenarios. We focus on a minimal use of computation, energy and storage resources at wireless sensors so as to address constrained devices: the proposed methods for key distribution and access control rely on extremely fast key derivation functions and, for the same reason, memory usage is reduced since keys are computed on the fly when needed. Our solution achieves privacy, authentication, semantic security, low energy, low computational demand and impacts mitigation of compromised devices on a simple manner. The access control provided is based on user identity and time intervals. We discuss these properties, compare our proposal to previous related work and provide experimental results that confirm its viability.

Key Words: access control, wireless network services, internet of things, sustainability

Category: C.2.4, E.3

1 Introduction

The paradigm known as the Internet of Things (IoT) defends the benefits of everyday objects becoming first-class citizens of the Internet. To do so, these objects must be provided with connectivity to expose and to consume data from any other applications or services. Nevertheless, the embedded devices used to connect the objects face different problems and challenges compared to normal computers. Due to the large scale of objects that will populate the IoT, these devices are usually designed to be small and inexpensive, resulting in limited processing capability. Additionally, these devices are often running 24 hours a day so low power consumption is required to enable sustainable computing.

Security-related routines sometimes impose an increment of energy consumption due to expensive calculations. Indeed, little attention has been paid to the security aspects in the IoT, and commonly security is left as a *dispensable*, energy draining process. The focus of this contribution is to define a novel and low consumption solution for restricting access to authorized users and securing communications among them and constrained devices. The solution enables a trustworthy communication on an insecure network at the cost of reduced energy consumption. The proposed model is compared with other solutions from the literature, showing how it is an advance in the state of the art on the field of efficient security with the restrictions commonly present in the IoT. Besides, experimental results are provided which show that the solution can be effectively implemented in practice. An earlier version of this work, without experimentation, was already shown in [Naranjo et al. 2012].

The contributions of this paper are twofold. First, a model is proposed to cover a typical scenario, adding the required access control layer, and second a review of the state-of-the-art and a comparison of this solution with different existing works is provided. Section 2 provides some needed cryptographic background, while Section 3 details the scenario we are addressing. Sections 4 and 5 introduce our proposal and show the experimental results, respectively. Finally, Section 6 compares the proposal with previous works focusing on several desirable features and Section 7 concludes the paper.

2 Background

In this section we provide a brief introduction to three basic cryptographic primitives that form the core of our proposal.

2.1 Message Authentication Codes

A Message Authentication Code (MAC) is a bit string unequivocally bound to a given message and symmetric key. MACs are used to assure the authenticity and integrity of messages between two communicating parties. A typical approach to MAC implementation is passing the message to a hash function and encrypting its output with the key. The receiving part then compares the decrypted hash to the output of a fresh hash operation on the received message and accepts the latter if the comparison is successful. Both parties should agree (maybe publicly) on either the hash and symmetric encryption algorithm used. They must also find a way of secretly sharing the symmetric key. The most popular scheme of this kind is HMAC [Bellare et al. 1996]. Other approaches for MAC implementation are based on block ciphers, like the DES-CBC MAC scheme [Dworking 05]. Computation of MACs is extremely fast at the cost of both parties needing to agree on a symmetric key.

2.2 Key Derivation Functions

Key Derivation Functions (KDFs) are used to derive proper symmetric cryptographic keys from a given secret input and (optionally) a public piece of information denoted as *salt*. It may happen that the secret input can not be directly used as a key because it does not fulfill the required properties of keying material or that more than one key is needed (e.g. one for encryption and other for authentication): in both cases KDFs provide arbitrarily long strong cryptographic material that can be used as encryption or authentication key. Other applications are key derivation from passwords (which are very low-entropy inputs) or even the generation of cryptographically suitable pseudo-random numbers. Let us remark that more than one key can be obtained from the same input by executing the function until enough bits are obtained.

Although KDFs are widely used there has been little effort towards standardization or theoretical security definitions. Instead, protocols and applications provide tailored solutions, valid for their specific requirements and circumstances. This is the case of those KDFs developed for TLS [Dierks et al. 2008], SSH [Ylonen et al. 2006] and many more.

However, two different general-purpose solutions have been recently proposed as standards: the NIST SP 800-108 [Chen 2008] and HKDF [Krawczyk 2010]. Both standards rely on interchangeable hash functions as their lowest-level building blocks and are therefore extremely efficient given that hash functions are based on bit permutation and transposition. Also, they claim to provide enough complexity to circumvent the fact that hash functions are not perfectly secure as sometimes assumed. Finally, they allow to input arbitrary long public information along with the salt. This public information may be user-dependent data such as a user identifier, for example. We refer the reader to [McGrew et al. 2010] for a nice survey on the matter.

2.3 Encryption in counter mode and semantic security

A symmetric encryption algorithm is semantically secure if, given two plaintexts and a ciphertext, an observer can not guess with a probability higher than a random choice (0.5) which of the two plaintexts produced the ciphertext. This property also receives the name "indistinguishability of encryptions", and it is important to prevent outsiders from guessing the contents of observed ciphertexts. The use of symmetric encryption alone[1] is considered a bad practice in cryptography since it does not provide semantic security: if two equal plaintexts are encrypted with the same key then both resulting ciphertexts are also equal, which would give hints to an attacker about what is being transmitted.

A typical way of achieving semantic security is using encryption in counter mode (CTR mode). For this, a counter is shared by the two communicating parties. When encrypting a block of data, the counter is encrypted and the result is exclusive-ored with the plaintext. The result is the ciphertext. The counter is incremented after every encryption, thus ensuring that different encryptions of the same plaintext will produce different ciphertexts. The counter can be public and there is no need to send it along with every ciphertext given that both parties know its current state. This technique was proposed in [Perrig et al. 2002].

3 Assumed infrastructure and cases of use

The infrastructure considered in our solution involves three kinds of players: sensors, Base Stations and user devices (e.g. smartphones).

Sensors are extremely constrained devices, frequently battery-powered and with reduced computational capabilities. They can be used for a wide range of tasks, like providing information based on measurable variables (e.g. temperature or humidity), surveillance of buildings, help in case of emergencies and many more. We assume they may also be connected to actuators in order to perform predefined actions related to physical access control (e.g. opening a gate to authorized users), ventilation (controlling the air conditioning system or opening windows after a temperature measurement), etc. Equipment and power shortage prevents sensors from performing the complex arithmetic operations involved in public-key cryptography in order to achieve encryption and authentication. However, symmetric cryptography is an option since many 802.15.4/ZigBee [IEEE 802.15.4-2006] compliant sensors have an Advanced Encryption Standard (AES) [FIPS 197] coprocessor installed.

Base Stations are better equipped devices that handle groups of sensors for message routing purposes and also for key management in our case. They are

[1] That is, Electronic CodeBook mode (ECB), i.e. encrypting different data blocks using the same symmetric key without additional security measures.

assumed to have a more powerful hardware and a permanent (or at least much longer) power supply and large storage space. They are also assumed to handle public-key cryptography routines and certificates.

Finally, users communicate with base stations and sensors through their smart devices, such as mobile phones, tablets, etc.

In order to illustrate the scenario we are focusing let us assume a home automation infrastructure. A set of sensors is deployed within a house, e.g. lights control, alarm or TV. Different users of the house may enjoy different access privileges and therefore can also be separated into different groups (e.g. adult owners, children, friends or relatives). Each group has a different set of permissions. For example, adult owners should have the highest privilege to access and control every single sensor/actuator; children may have access to the TV actuators, but won't be able to purchase pay-per-view programs; and friends will be able to access the WIFI and turn on and off some lights of the house. These permissions are issued by members of the adult owners group.

4 Our proposal

Our main goal is to allow sensors and legal user devices only to establish encrypted and authenticated one-to-one channels while minimizing the intervention of the Base Station. The process should require a small amount of energy consumption and storage, specially on the sensor side. Minimizing storage requirements also implies that communications should be as stateless as possible, i.e., no inter-session information should be stored for long periods of time. Besides, it should be easy for the sensor to perform access control operations on user devices.

Our solution covers four phases: sensor bootstrapping, user join, regular communication and user eviction, all of which are described next. For the sake of simplicity and without loss of generality we focus on a simple scenario: one Base Station (namely BS), one sensor (namely S), and one user device (namely A). The extension of the proposed protocol to several users, sensors and base stations is straightforward from the protocol description below. Messages involved in the protocol are depicted in Figure 1 while Table 1 shows the notation used from now on.

Sensor bootstrap. At the time of adding the new sensor S, the BS generates a master secret MS_S. This secret is sent to S under a secure channel[2].

[2] There are a number of solutions in the literature for the problem of creating a secure channel between two devices in a wireless network of constrained devices. [Zhang et al. 2010] thoroughly surveys the matter.

In our case, the simplest solution is the manual installation of a symmetric key both in S and BS before deployment. That key can be used to send MS_S encrypted.

A more elaborate solution can be found in [Zhu et al. 2006]. Its authors propose to

MS_S	Master secret for sensor S
$Kenc_{S,A},\ Kauth_{S,A}$	Encryption and authentication keys for communication between sensor S and user A
$Kenc_{S,A}\{x,\ ctr\}$	x is encrypted in counter mode using key $Kenc_{S,A}$ and counter ctr
$MAC_{Kauth_{S,A}}(x)$	A MAC is done on x with $Kauth_{S,A}$
$KDF(x,\ \{a,\ b\})$	A Key Derivation Function is applied to master secret x using a as public salt and b as user-related information
$H(x)$	A hash function is applied to x
$x\|\|y$	Concatenation of x and y
ID_A	Identifier of user A
a	Random integer salt
$init\_time,\ exp\_time$	Absolute initial and expiration time of a given key

Table 1: Notation

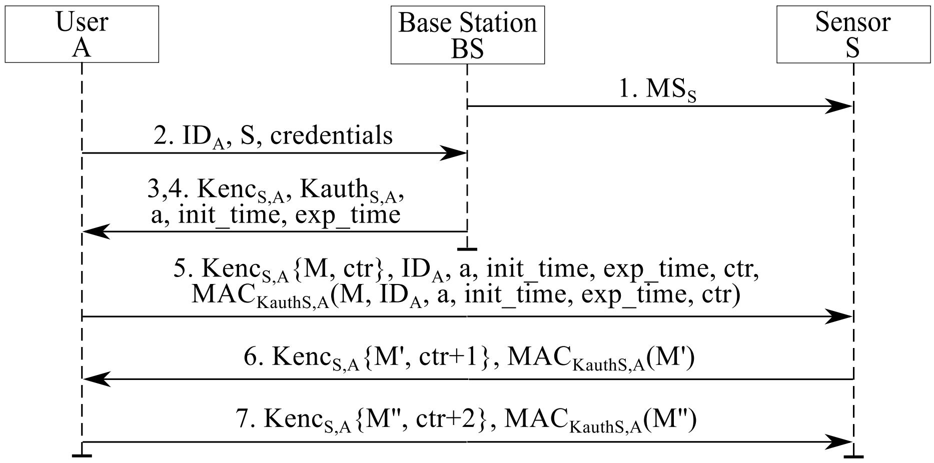

Figure 1: Messages involved in the protocol

1. $BS \rightarrow S$ (secure channel) : $[MS_S]$

User join. Let us assume that S is already operating under normal conditions. User A arrives at the scenario handling her mobile device and wishes to request some information from S. First, A sends a request to BS asking for keying material to communicate with S (step 2). The message should include authentication and authorization information so BS can perform high-level access control on user A. For this we suggest the use of public key cryptography [ITU-T X.509]

install a common secret in all sensors and the BS. After deployment, each pair BS-S agrees on a symmetric key known only by them. This key is generated from the initial common secret and the identifiers of both BS and S by means of a pseudo-random function, and can then be used to send MS_S encrypted.

given that *(i)* both the Base Station and the user device are assumed to handle it easily and *(ii)* it also allows to create an ad-hoc secure channel between them. In any case, let us remark that this step is only performed at user arrival and that many target sensors can be requested in the same message at step 2.

2. $A \rightarrow BS:\ [ID_A,\ S,\ credentials]$

If A's request is accepted then the Base Station generates appropriate keying material for every sensor indicated (step 3, here we show keying material for one sensor only, S) and sends it to A through a secure channel (step 4). The expiration time of this material is decided by the Base Station and cannot be changed by A.

3. BS computes:
   (a) a, random integer salt[3]
   (b) $(init\_time,\ exp\_time)$, keying material validity interval
   (c) $Kenc_{S,A},\ Kauth_{S,A} = KDF(MS_S, \{a,\ ID_A||init\_time||exp\_time\})$
4. $BS \rightarrow A$ (secure channel) : $[Kenc_{S,A},\ Kauth_{S,A},\ a,\ init\_time,\ exp\_time]$

Regular communication. A can now use the received keying material to encrypt and authenticate her first message M addressed to S. For this, A uses a fresh random counter *ctr* and sends it along the message with the rest of public user-dependant information.

5. $A \rightarrow S : [Kenc_{S,A}\{M,\ ctr\},\ ID_A,\ a,\ init\_time,\ exp\_time,\ ctr,$
   $MAC_{Kauth_{S,A}}(M,\ ID_A,\ a,\ init\_time,\ exp\_time,\ ctr)]$

Upon reception of the message, sensor S computes the corresponding keying material as in step 3c. S can now decrypt and authenticate the whole message. Before encrypting the reply message M', S increments *ctr*. No additional information is needed (step 6).

6. $S \rightarrow A : [Kenc_{S,A}\{M',\ ctr + 1\},\ MAC_{Kauth_{S,A}}(M')]$

The counter is incremented by the corresponding sender in every subsequent message. If, for any reason, any of the players loses synchronization regarding the counter it can always recover it by trying consecutive values until the proper one is found. This resynchronization process should not take long for short message exchanges. The same technique is used by the well known SNEP protocol [Perrig et al. 2002].

[3] We assume that MS_S and a are obtained from a secure pseudorandom number generator.

When the message exchange finishes the sensor can delete the keying material related to A since it can be easily recomputed in the next exchange. The protocol therefore does not require S to store any information about A between two message exchanges.

Finally, we have considered so far the necessity of privacy between A and S. If the service provided by S does not require privacy then messages do not need to be encrypted, just authenticated with $Kauth_{S,A}$ (this is applicable to the rest of the paper).

User eviction. The inclusion of a validity time interval in the key derivation function input provides easy time-based access control. Before computing the keying material after step 5 the sensor S checks whether the expiration time has not yet been reached. This needs a very relaxed time synchronization with the BS, in the order of seconds, while other well known protocols impose much stronger requirements on this matter (centiseconds or milliseconds) [Perrig et al. 2002, Chowdhury et al. 2011]. Also note that A cannot fake her ($init\_time$, exp_time) pair because the keys derived by the sensor will be different and communication will be impossible. Consequently, the user is *forced* to be honest. Finally, the Base Station may decide to evict A before her expiration time in certain situations, e.g. due to misbehaviour or key exposure. This is more problematic: the only way of making S reject messages from A before exp_time is to maintain a blacklist with (ID_A, exp_time) items in every sensor, which requires an additional communication per item between the BS and the sensor. However, the scarce storage space at the sensor will not allow for long blacklists. Fortunately, items can be removed as soon as their corresponding expiration times are surpassed: from that moment on sensor S will reject step 5 messages basing on the attached obsolete exp_time value rather than on ID_A.

4.1 Considerations on security

No keys are publicly disclosed, nor they even travel encrypted in the user-sensor message exchange. Following good cryptography practice, different keys are used for encryption and authentication so the use of a single key for more than one task is avoided. The impact that a sensor compromise makes on the rest of the network is reduced since there are no shared keys among sensors nor among users: every sensor owns a different master secret pair, so an attack on that node would not provide any knowledge about other sensors in the network. In a similar way, each user knows only those keys shared with a given set of sensors and, what is more, those keys are exclusive for her. This means that a compromise on them would only allow to impersonate that user. This is not the case of [Ngo et al. 2010] (see Section 6).

that h must be communicated in step 5 and that different key chains may overlap (e.g., [h, h+15] and [h-10, h+5]). To avoid the latter the user can choose ever-increasing, sufficiently scattered values for h.

Semantic security is achieved thanks to the use of counter mode encryption as stated above: if the counter is updated after every message then different encryptions of the same message will produce different ciphertexts. As a final remark on this, note that the user decides on the initial value of ctr at the beginning of every message exchange. If the sensor does not trust users by default then it can choose a different value on step 6, say ctr', and include it in the response message. The user should then use $ctr' + 1$ in the next message and so on. In any case, the sensor can delete this information along with the user-specific keying material when the transaction ends.

4.2 Considerations on overhead and storage

Very little overhead is added to message length in user-sensor communications: the user just needs to attach (a, $init\_time$, exp_time, ctr) in step 5 (we assume that ID_A must be sent anyway). The additional information sent by the weakest player, i.e. the sensor, is minimal: only ctr' in step 6 if desired, nothing otherwise. This short message overhead helps to reduce the energy used by the sensor when transmitting (by far the most energy-consuming operation in sensors).

The permanent storage requirements imposed on the sensor are also extremely reduced. It only needs to store *(i)* a symmetric session key for communications with BS and *(ii)* the master secret MS_S. While exchanging messages with user A the sensor stores *(i)* the last received message M, which includes (ID_A, a, $init\_time$, exp_time, ctr), and *(ii)* ($Kenc_{S,A}$, $Kauth_{S,A}$). Any available space left can be used for a user blacklist if desired. On the other hand, the fact that the user must keep a pair of keys per sensor might be seen as a downside. However, current smartphones and similar devices have huge long-term memories in comparison to sensors.

5 Experimental results

We have implemented our proposal in Java and tested it on a Raspberri Pi unit [Raspberri Pi], a low cost single-board device with a 700 MHz ARM11 processor. The code carries out the computations stated in step 3c of the protocol in Section 4, which is the core of our proposal and its most resource-demanding stage. Note that both $Kenc_{S,A}$ and $Kauth_{S,A}$ are computed in every execution. The NIST standard [Chen 2008] was used as key derivation function (KDF). Without going into detail, NIST calls a pseudo-random function which is not specified in the standard. For this we chose the popular and secure HMAC with SHA-256 [Bellare et al. 1996][FIPS 180-4] which produces 256 bits outputs. Those bits are in turn used to generate the output key material. Table 2 shows the results.

	Output key size		
	128 bits	**256 bits**	**512 bits**
Exec. time (ms)	12.2	12.2	22.4
Energy consumed (mJ)	3.3	3.3	5.1

Table 2: Experimental results for NIST with different output key sizes.

Average execution times for step 3c and different key lengths were obtained (every experiment was repeated 1000 times). One can see that the time for 512 bits nearly doubles the others. This is due to the use of SHA-256: for 128 and 256 key lengths only one call to SHA-256 is required since enough bits are provided. However, for a key length of 512 bits the algorithm needs to call SHA-256 twice. This fact clearly shows that the pseudo-random function (in our case HMAC with SHA-256) consumes most part of the execution time. Therefore a clever selection of that function based on the desired key length is a crucial issue.

Energy figures show the average additional energy consumed by an execution of step 3c. In a steady state, the Raspberri Pi consumes an average of 37.6 mJ during a similar amount of time (to obtain the total consumption the reader should add this last figure to those of the table). The execution therefore drains only an additional 8.77% of the steady state's consumption in the 128 and 256 bits cases, and an additional 13.59% in the 512 bits case.

6 Related work

SPINS [Perrig et al. 2002] provides lightweight symmetric encryption and authentication in WSNET scenarios in which a Base Station is actively involved. It is composed of two different protocols: SNEP and μTESLA. The SNEP protocol provides encryption, authentication and data freshness evidence between two parties, using symmetric encryption in counter mode to ease freshness verification and to thwart replay attacks. On the other hand, the μTESLA protocol provides symmetric authentication: the sender builds a hash-based key chain and discloses keys with and intentional delay. Based on that delay and on the one-way property of hash functions, message recipients can verify the authenticity of messages.

LEAP+ [Zhu et al. 2006] proposes an authentication and encryption framework for the same scenarios addressed by SPINS: wireless networks of sensors communicating among them and with the Base Station. Apart from its own protocols, μTESLA is used to provide broadcast communications by the Base Station. LEAP+ is proven to be lightweight and low demanding in terms of energy

	SPINS	**LEAP+**	**Ngo**	**MAACE**	**Ours**
Number of keys per sensor	1 for BS	1 for BS 1 per neighbour 1 for cluster	1 for group	1 for BS	1 for BS 1 for basic
Number of keys per user	-	-	2	1 for BS	2 per sensor
Tight clock synchronization	Yes	No	No	No	No
BS is highly involved	Yes	No	Yes	Tradeoff	No
Limits impact of sensor compromise	Yes	Yes	No	Yes	Yes
Services scenario	No	No	Yes	Yes	Yes

Table 3: Feature comparison. LEAP+ is considered without μTESLA.

consumption. Its cornerstone operation is an interesting proposal for limiting the impact of a single node compromise based on pseudo-random functions.

Ngo et al [Ngo et al. 2010] proposed an access control system for the scenario we address here: wireless networks that provide services to users. Either service and user groups are supported with the help of an Authorization Service. Two computationally lightweight protocols are described: while the first one allows to prove a user group membership to a service, the second protocol is intended to authenticate against a service, both individually and per-group, by employing the user's individual key and the group key. Note that a stolen group key would compromise the whole group.

The very recent MAACE [Le et al. 2011] addresses the same scenario and is very similar to our proposal according to the key generation process and the security achieved. User-Base Station communications are secured with public-key cryptography, while user-sensor messages are encrypted with a newly established symmetric session key. However, two drawbacks can be found in this scheme. First, the session key shared with the sensor is chosen by the user, which is not a desirable feature (she might use a deliberately weak key). In our scheme, the Base Station takes care of this task. Second, the sensor must store all keys shared with online users at a given time (which requires a large storage space), or involve the trusted device in frequent communication establishments (which would make the process less efficient). In our scheme, the sensor can generate any valid key on the fly.

Table 3 compares our proposal to the reviewed related work according to some relevant features. All protocols shown provide encryption and authentication.

7 Conclusions

This work introduces a simple user access control solution for wireless network services in IoT scenarios. The typical infrastructure of these scenarios is composed of Base Stations and sensors, with users interacting through their smart devices (e.g. mobile phones). We focus on a minimal use of computation, energy and storage resources at the sensor so as to address constrained devices: key distribution and access control rely on extremely fast key derivation functions and, for the same reason, memory usage is reduced since keys are computed on the fly when needed. This way, adding security to an IoT scenario does not imply a high energy consumption which would disable sustainability. Our solution provides encryption, authentication, semantic security and access control based on user identity and time intervals without requiring tight clock synchronization among devices. Besides, the intervention of the Base Station in user - sensor communications is minimal, which is also a desirable feature. Experimental results prove that the solution can be effectively implemented in real scenarios. Regarding future work, we expect to test our solution experimentally in more constrained devices such as wireless motes like MICAz or the Arduino platform. Additionally, extending the protocol to manage groups of users and sensors seems to be an interesting research direction.

Acknowledgments

This work was funded by the Spanish Ministry of Economy and Competitiveness (TIN2008-01117, TIN2012-37483-C03-03, IPT-2011-1558-430000 and TIN2012-37483-C03-03), Junta de Andalucía (P11-TIC-7176), and partially by the European Regional Development Fund (ERDF).

References

[Bellare et al. 1996] Bellare, M., Canetti, R. and Krawczyk, H. "Keying Hash Functions for Message Authentication". In Proceedings of CRYPTO (1996) pp: 1-15, Springer.

[Chen 2008] Chen, L. "Recommendation for Key Derivation Using Pseudorandom Functions". NIST Special Publication 800-108 (2008).

[Chowdhury et al. 2011] Chowdhury, A. R. and Baras, J. S. "Energy-Efficient Source Authentication for Secure Group Communication with Low-Powered Smart Devices in Hybrid Wireless/Satellite Networks". EURASIP J. Wireless Comm. and Networking (2011).

[Dierks et al. 2008] Dierks, T. and Rescorla, E. "The Transport Layer Security (TLS) Protocol Version 1.2". RFC 5246 (2008).

[Dworking 05] Dworkin M. “Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication”. NIST Special Publication 800-38B (2005).

[FIPS 197] Federal Information Processing Standard 197. “The Advanced Encryption Standard”. NIST (2001).

[FIPS 180-4] Federal Information Processing Standard 180-4. ‘Secure Hash Standard (SHS)”. NIST (2012).

[Goldreich et al. 1986] Goldreich, O., Goldwasser, S. and Micali, S. “How to construct pseudorandom functions”. Journal of the ACM (1986) 33:4, pp: 210-217.

[IEEE 802.15.4-2006] 802.15.4-2006 IEEE Standard for Information Technology - Part 15.4. “Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPANs” (2006).

[ITU-T X.509] Information technology - Open Systems Interconnection - The Directory. “Public-key and attribute certificate frameworks. ITU-T recommendation X.509 (2005).

[Krawczyk 2010] Krawczyk, H. “Cryptographic extraction and key derivation: the HKDF scheme”. In Proceedings of CRYPTO (2010) pp: 631-648, Springer.

[Krawczyk et al. 2010] Krawczyk, H. and Eronen, P. “HMAC-based Extract-and-Expand Key Derivation Function (HKDF). rfc 5869 (2010).

[Le et al. 2011] Le, X.H., Khalid, M., Sankar, R. and Lee S. “An Efficient Mutual Authentication and Access Control Scheme for Wireless Sensor Networks in Healthcare”. Journal of Networks (2011) 6:3 pp: 355-364.

[McGrew et al. 2010] McGrew, D. and Weis, B. “Key Derivation Functions and their Uses”. IETF Draf (2010).

[Naranjo et al. 2012] Naranjo, J.A.M., Orduña, Pablo, Gómez-Goiri, Aitor, López-de-Ipiña, Diego and Casado, L.G. “Lightweight User Access Control in Energy-Constrained Wireless Network Services”. In “Ubiquitous Computing and Ambient Intelligence”; Lecture Notes in Computer Science, (2012) pp: 33-41. Springer.

[Ngo et al. 2010] Ngo, H.H., Xianping W., Phu D.L. and Srinivasan, B. “An Individual and Group Authentication Model for Wireless Network Services”. JCIT (2010) 5:1, pp: 82-94.

[Perrig et al. 2002] Perrig, A. and Szewczyk, R. and Tygar, J. D. and Wen, V. and Culler, David E. “SPINS: security protocols for sensor networks”. Wireless Networks (2002) 8:5, pp: 521-534.

[Raspberri Pi] The Raspberri Pi platform. `http://www.raspberrypi.org/`

[Ylonen et al. 2006] Ylonen, T. and Lonvick, C. “The Secure Shell (SSH) Transport Layer Protocol”. RFC 4253 (2006).

[Zhang et al. 2010] Zhang, J. and Varadharajan, V. “Wireless sensor network key management survey and taxonomy”. Journal of Network and Computer Applications, (2010) 33:2, pp: 63-75.

[Zhu et al. 2006] Zhu, S., Setia, S., Jajodia, S. “LEAP+: Efficient security mechanisms for large-scale distributed sensor networks”. ACM Transactions on Sensor Networks, (2006) 2:4, pp: 500-528.