An Authentication and Access Control Framework for CoAP-based Internet of Things

Pablo Puñal Pereira, Jens Eliasson, Jerker Delsing
Dept. of Computer science, Electrical and Space Engineering
Luleå University of Technology
Luleå, Sweden
Email: pablo.punal@ltu.se

Abstract—Internet of Things (IoT) and Cyber-physical Systems (CPS) are two very hot research topics today, and more and more products are starting to appear on the market. Research has shown that the use of Service Oriented Architecture (SOA) can enable distributed application and devices to device communication, even on very resource constrained devices, and thus play an important role for IoT and CPS.

In order to realize the vision of Internet of Things, communication between devices must be secured. Security mechanisms for resource constrained devices has attracted much interest from the academic community, where research groups have shown solutions like IPsec, VPN-tunnels, (D)TLS, etc. are feasible to use on this type of networks. However, even though the use of well-known security mechanisms are vital for SOA-based IoT/CPS networks and systems to be protected, they do not provide any fine-grain access control.

In this paper, a CoAP-based framework for service-level access control on low-power devices is presented. The framework allows fine grain access control on a per service and method basis. For example, by using this approach a device can allow read/write access to its services to one group of users while only allowing read access to another group. Users without the right credentials are not even allowed to discover available services. To demonstrate the validity of the proposed approach, several implementations are presented together with test results.

The aim is to provide a holistic framework for secure SOA-based low power networks comprise by resource constrain devices.

I. INTRODUCTION

The use of Service-Oriented Architecture (SOA) on resource-constrained devices has gain a lot of interest from both the academy as well from the industry in recent years as shown by [1], [2]. Service-oriented Architecture is based around the notion of services, formal interfaces and standardized protocols. A service is the core building block of SOA, and is piece of software performing some task, encapsulated with a formal interface described using some standard description format such as WSDL, and WADL in the case of Web services. A service must hold certain properties, such being discoverable, composable and loosely coupled from any operating system, programming language and other services. Since services are distributed in their nature, and relies on communication channels to exist to function, they are inherently vulnerable for issues such as hackers, malware and other network based intrusion threats, for example denial of service (DOS) attacks. Services must therefore be protected

using communication protocols using strong encryption and authentication, such as IPsec [3], SSL and other mechanisms. If Internet of Things and Cyber-physical Systems are to become mainstream technologies, used by millions of users, there must exist strong communication security and reliable authentication mechanisms. For example [4], Kasinathan et al. investigated mechanisms for detecting denial-of-service (DOS) attacks, which can be used to disrupt a network. 6LoWPAN is especially sensitive of DOS attacks due to the low bandwidth.

Regarding security, the utilization of IPsec (IP Security) over low-power networks was increased during the last years. IPsec has two modes of operation: the Transport Mode, which adds a Authentication header between the IP header and the UDP/TCP header, which allows the system to validate incoming packets but the original data is visible and accessible for all other devices on the network. The second mode is called Tunnel Mode, which is similar to the first one except that the IP header, the UDP/TCP header and the payload are encapsulated and encrypted (typically using AES) as payload of a new IP packet. This mode protects packets against eavesdropping attack, discards data modification and adds the possibility to detect Denial of Service attacks.

IPsec needs a shared password to encrypt and decrypt properly all incoming and outgoing messages. If these passwords are static could be compromised after some thousand messages. To solve this problem the IKE (Internet Key Exchange) and IKEv2 protocols were created. These protocols guarantee a safety communication between two devices and are able to create new shared passwords using circling derivative methods.

To protect UDP packets (even over IPsec), there is another protocol that can be used to add an extra layer of protection called Datagram Transport Layer Security (DTLS) [5]. This protocol uses a initial handshake to set the passwords. After that the content of the UDP packet is encrypted (usually with TLS_PSK over AES) and a header of 13 bytes is added, together with the initialization Vectors (IV) (over 8 bytes for AES128), integrity values (8 bytes) and the padding required by the cipher suite. DTLS increases the size of the packet, but this is a consequence of the packet encapsulation. This protocol is fully integrated in the CoAP protocol [6].

However, even though the use of well-known security mechanisms such as IPsec, VPN-tunnels, SSL, (D)TLS etc are vital for SOA-based networks and systems to be protected,

they do not provide any fine-grain access control mechanisms. For example, if computers are exchanging data using a SOA-enabled protocol such as CoAP, only the packets are protected from external tampering and access. Any request from a client already inside a protected tunnel will be accepted by the server. The use of DTLS-encrypted CoAP could be one way of achieving a per service access control, but this would require a large number of different key-pairs to be in use in order to enable a true access control where a client can have different security access to different services, and even access to different operations on one service. For example, a client can have GET permissions to an actuator in order to view the status, but might not be allowed to actually perform a change of state using PUT or POST on that service. If only a few DTLS keys would be used, then fine-grain access is not possible. If a very large number of keys are used, then fine-grain access control is possible but the management and administrating of key exchange mechanisms would be difficult.

A better approach is to separate the access control from the communication security. This would increase the security since another layer of protection is added. A client would also only need a few keys for the IPsec and DTLS encryption, and then use an authentication service to gain access to other services. This makes administration easier since all access rights are centralized in the authentication service. Two access control protocols that use this approach are Kerberos [7] and RADIUS [8]. Kerberos is an authentication protocol which works on the basis of 'tickets' to allow nodes communicating over a non-secure network to prove their identity to one another in a secure manner. RADIUS is often used for network authentication in wireless domains, and supports Access control, Authentication and Accounting (AAA).

This paper proposes a CoAP-based framework that solves the problem of a fine grain access control, which is not possible with other connection control system like IPsec and DTLS. The framework is focused on low overhead on resource-constrained devices that are commonly used in network for Internet of Things and Cyber-physical systems. The proposed solution uses ideas for other access control systems like Kerberos and RADIUS, and merge the two with the CoAP protocol to get a reliable access control framework for IoT.

This paper is structured as follows: Section II presents the background and related work, followed by a presentation of the proposed architecture in Section III. After comes Section IV which provides a detailed presentation of the authentication process, followed by a security analysis in Sec. V. Section VI outlines the performed experiments and results. Finally, future work and the paper's conclusions are presented in Sections VII and VIII, respectively.

II. BACKGROUND AND RELATED WORK

In this section, the background and the reason of this work are described, with a CoAP protocol description and some authentication protocols, and methods that were a base for the proposed framework.

A. Industrial networked devices and security

Industrial usage of networked for automation has been around for a long time. The industry is healthy with expected growth of 7% or more [9]. The projected big numbers of connected devices [10] indicate an even more rapid growth in networked devices for industrial usage automation. The discovery of the Stuxnet virus [11] and the information from the whistle blower Edward Snowden opened the eyes of the industry and general public about that any connected device might be vulnerable to Internet and electronics security issues. Currently much effort is devoted to prevent and protect against cyber attacks on networked devices. This certainly is true for Internet of Things.

In the field of networked resource constrained devices certain protocols are gaining popularity. Some of the most interesting ones, from the security point of view, are briefly reviewed below.

B. CoAP (Constrained Application Protocol)

The IETF Constrained Application Protocol is an application-layer protocol designed to provide web services working with constrained nodes. The protocol is designed for low-power networking. CoAP provides a request/response interaction model between application end-points, supports built-in discovery of services and resources, and includes key concepts of the Web such as URIs, RESTful interaction, extensible header options, etc. CoAP easily interfaces with HTTP for integration with the Web while meeting specialized requirements such as multicast support, very low overhead and simplicity for constrained environments. CoAP uses UDP unlike HTTP. Some features of CoAP are:

- Two types of request messages: Confirmable Message (CON) - the message is retransmitted (four times maximum) with an exponential time out waiting for an Acknowledged Message (ACK) or the correct response from the server. The second type is the Non-Confirmable Message (NON) - the message is sent without any kind of response.
- The URI format allows the use of standard and specialized service endpoints. One for example is the resource discovery defined in RFC 5785 [12] that uses the *well-known/core* path and the CoRE link format.
- CoAP also allows to send very big messages with a stop-and-wait mechanism called "blockwise transfers" (splitting messages and sending them with a reference order).

0								1								2								3																																																																							
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1																																																																
V				T				TKL				Code								Message ID																																																																											
Token (if any, TKL bytes) ...																																																																																															
Options (if any) ...																																																																																															
1																1																1																1																1																Payload (if any) ...															

Fig. 1. Original CoAP packet format.

The CoAP packet format (see Figure 1) has a maximum length of 1400 bytes, but the header has a length of 32 bits (2

for the version control, 2 for message type, 4 for token length, 9 for the message code and 16 for the message ID).

0							
0	1	2	3	4	5	6	7
Delta				length			
Delta (extend) 0 - 2 bytes							
length (extend) 0 - 2 bytes							
Value 0 - ...							

Fig. 2. CoAP option format.

C. Kerberos

Kerberos [7] is a protocol that uses a primary communication between the client and the Authentication Server (AS) to generate a valid ticket. This ticket will be used for future accesses to Service Servers (SS). There are Kerberos implementations that run over UDP or TCP. Also the Ticket generation process could include different encryption methods, everything is flexible and configurable by the network administrator.

D. RADIUS (Remote Authentication Dial In User Service)

RADIUS [8] is a networking protocol to provide Authentication, Authorization and Accounting management centralized in a single server. This protocol offer the possibility to configure a single Network Access Server (NAS) into a user specific NAS. The use of it is widely used. It was designed in 1991 and allows many different types of configurations, but always work over UDP. This protocol supports Challenge responses (as PAP and CHAP) increasing the security against Eavesdropping attacks and also supports the use of certificates like X.509 [13]. The RADIUS packet format is shown in Figure 3.

0								1								2								3							
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Code								Identifier								Length															
Authenticator																															
Attributes (if any) ...																															

Fig. 3. Original RADIUS packet format.

There are only five possible fields: Code field (to identify request/response type), Identifier (to identify each packet), Length (to know the packet size), Authenticator and Attributes. The first four are always required, but the last one is optional. The Authenticator is a field of 16 bytes that is used to send encrypted data only to trust the communication between server and client. The encryption type is not predefined, originally this protocol was designed to use MD5 [14] but nowadays is not sufficient, therefore any other HASH generator could be used instead of MD5, like SHA-512 [15] or PBKDF2 [16]. After a request, there are three types of responses:

- Access Reject: the user has not access to any network resources. The reason could be a failure or a wrong identification.
- Access Challenge: to increase the security, the server could request extra information before trust on that user.
- Access Accept: The user has granted access.

The RADIUS protocol and format could run over CoAP protocol (see section IV-A), because all connection features can be possible also over CoAP. Also it is useful to authenticate devices over 6LoWPAN and enable the access to the network (see future work at section VII). The overhead of this protocol affects only to the authentication process (see section IV and Fig. 6). Therefore it is reasonable to use in low-power and low band width networks.

E. Diameter

Diameter [17] is the evolution of RADIUS, the biggest modification is the use of TCP/SCTP instead of UDP. This change increase the security performance as much as TCP allows error handling, capability to negotiation and alive connections. But it also makes Diameter non-backwards compatible with RADIUS, and also with CoAP.

III. FRAMEWORK

In this section all components of the proposed architecture are explained, with their ability to provide robust low power authentication and access control mechanisms.

A. Requirements

In order to get sufficiently fine grain access control, the framework must be designed optimal in terms of computation and overhead for access control and security over low power devices and networks with low width band. The communication to the access control service must be enable for all network devices, at least for valid devices. Is possible to filter the connection to access control service by IPsec and DTLS. The access control must be independent of other devices (there are no relation between two different accesses) and must be easy to automatize, due to most of these will be done by machines. The access rules and methods could be different depending on authentication polices, witch could change in base of local conditions, like time, power source, position, sensor information, etc. The final objective is reach a fine-grain access control per service (that includes access methods and specific tags).

B. Scope

The aim of the proposed framework is to provide fine-grain and low-power authentication and access control mechanisms for resource-constrain devices, however the framework does not focus on communication security but relies on existing secure protocols such as IPsec and DTLS (see Fig. 4).

The lowest communication security level is based on IPsec (see Figure 4). The implementation of this IPsec layer is based on the open source implementation by Raza [18].

Fig. 4. Communication layers

C. Access Control and Authentication Aspects

The proposed framework uses the best benefits of CoAP, Kerberos and RADIUS solutions to create a low-power platform for AAA. The propose architecture is shown Figure 5.

Fig. 5. Simplified architecture

The Client must be a CoAP client ready to get the ticket after a login and use it in each future CoAP request. The AAA server checks the authentication requests and communicates the result to the CoAP server. CoAP server has the following requirements:

- Must check all the incoming packets looking for a CoAP Option [#100] called Ticket (Kerberos approach). This Ticket must validate the user and must give permissions to use the specific service by the user. All incoming request without Ticket will be discarded except Reset (RST) and Acknowledge (ACK) messages. Optional the [.well-known/core] could be also discarded to protect the network against automatic attacks.
- The authentication service [.well-known/auth] must exist on the CoAP server.
 - Must have two different options [.well-known/auth?login] to do the login and [.well-known/auth?logout] to do the logout.
 - If the login process fails, the server must send an error message. If the login process succeeded, the server must generate and send a new ticket together with the timeout.
 - The logout process must delete the ticket on the server side and send a message to the user that the logout was successfull.

IV. AUTHENTICATION PROCESS

The Authentication and Access Control consists of two different steps. The first step is a Authentication. In this step the system must recognize user as a valid user (with a password, a shared key or some other validator). This process will inform the CoAP-NAS about who the user is, permissions, group, time out of the ticket, etc. At the end of this process the CoAP-NAS must send the user a valid ticket but only if the authentication process is successful. In other case the server must inform the user that the authentication is not valid for the authentication process (incorrect user, incorrect password, black list, etc.). The second step is the Access control. On each client request, a valid ticket must be included to identify this request to the valid user, then the server will recognize the user and will respond with the correct message. If the client does not send the ticket or a valid one, the server will respond with an error message. The complete process is shown in a scheme in Figure 6.

Fig. 6. Authentication process

A. Authentication Method

On the authentication process the server must recognize the user as a valid user and communicate that to the CoAP-NAS. This process needs to be flexible and compatible with other standards and with this goal the propose framework creates a public login CoAP service on the CoAP-NAS. This login service must receive a PUT request with one of the following contents as a payload:

- User name and password as plain text. This option is only recommended during testing, debugging and development phases.

- User name and password hash. This is easy to implement and could be authenticated directly on the CoAP server (without RADIUS).
- A RADIUS packet (future work).

The possibility to run RADIUS protocol over CoAP (see section II-D) gives to the framework a flexible authentication method usable with a standard RADIUS server. This approach requires no RADIUS protocol on the client, then the overhead and the required resources will be smaller compared with the use of both protocols at the same time. This is especially important for resource-constrained sensor nodes. Therefore, the conversion between RADIUS packet and CoAP-RADIUS packet is simple. In this framework there are proposed two alternatives, the first one is the most compatible with RADIUS standard, due to all of a RADIUS packet will be in the CoAP payload packet (Figure 7). The alternative option is omit any

0								1								2								3															
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1								
V		T		TKL				Code								Message ID																							
Token (if any, TKL bytes) ...																																							
Options (if any) ...																								1		1		1		1		1		1		1		1	
Code								Identifier								Length																							
Authenticator																																							
Attributes (if any) ...																																							

Fig. 7. RADIUS over CoAP packet

redundant information as much as possible, thus deleting the Code, Identifier and Length in the RADIUS packet. This could be directly translated from the CoAP ID and Code (shown in Figure 8).

0								1								2								3							
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
V		T		TKL				Code								Message ID															
Token (if any, TKL bytes) ...																															
Options (if any) ...																															
Authenticator																															
Attributes (if any) ...																															

Fig. 8. Compressed RADIUS over CoAP packet

B. Control Method

The servers could manage a lots of connections per second and CoAP protocol is focused on communication with low overhead packets. Then, the access control process must not increase too much the normal size of the packets. For this reason the propose framework would add a new option into the CoAP standard option set. This option called Ticket has the following properties:

- Unique per user and session. When a session expires the user must authenticate again and will receive a new one. The use of a time out will further increase the security.

- The ticket length could be configurable from 32 to 128 bits according to the requirements.
- Alternative to increase the security level, the first Ticket could be used as a seed to create a new one on each communication. This dynamic generations must be done with the use of hashing techniques.

On each incoming packet, the server must know from the CoAP packet the Message ID, the Ticket, the Service Name or URI to access and the Method. From the IP packet the server knows the IP address and the port number. For each authentication process the server must keep on memory the user name, the user password (or a hash), the IP address, the port number, the generated Ticket and the time stamp for time out. With all this information the server is able to recognize the user and check the permissions in a database. This process must done with the highest priority to detect a valid or non-valid ticket. At this point, if the ticket is valid, the server must send a normal response (according to the CoAP specifications). If the ticket is wrong or there is no ticket on the packet, the server must send an error message with the error code 406 (Not Valid) to inform the user that has not permissions to use that service. The propose framework is able to detect if a client is sending wrong tickets to ignore it.

V. SECURITY ANALYSIS

This section shows the security features of the proposed platform against different attacks.

- 1) Eavesdropping attack: IPsec and DTLS protect the system against this type of attacks. But to increase the security, the proposed platform could generate a new ticket for each message (plus hashing it with other parameters like Message ID), thus increasing the difficulty for a malicious user to predict a valid ticket.
- 2) Data modification attack: IPsec layer protects against this type of attacks. If the data is modified the Authentication Header of IPsec will detect that.
- 3) Man in the middle attack: IPsec will encrypt the data and will use time outs for each IP connection (with IKE), increasing the difficulty to guess the valid password to decrypt all packets.
- 4) Identity Spoofing attack: the use of false IP addresses is not going to work over the IPsec layer.
- 5) Denial of Service attack: there is no protection against this type of attacks. It is possible to decrease the overload discarding all dirty packets directly on the IP layer (using IPsec), but finally if the attack is severe the system will finally collapse (see [4]).
- 6) Application-Layer attack: this is considered the most difficult type of attack. In this case, the IPsec layer and (D)TLs are not going to protect the system. Here, only services with are under access control will be protected.
- 7) Replay attack: IPsec layer protect those attacks with the use of sliding windows.

The security features depend directly on the complexity of the encryption method and also on the length of the keys.

The proposed framework do not need a specific cipher suite or a hash generator, but the security performance will depend directly of this selection.

VI. EXPERIMENTS AND RESULTS

To perform the experiment, the server is based on libcoap 4.1.1 [19]. The source code of the server was modified to integrate the management of the users and the groups on each service with the permissions. The Copper (Cu) Firefox add-on [20] was used as a client. This enables the latest version of CoAP (draft 18) to be used, and has a user-friendly Graphical User Interface. The Copper plugin needed to be modified in order to integrate the ticket management and to add a visual user login menu (shown in Fig. 9).

Fig. 9. Copper login menu

To test the proposed fine-grain access control, several services were created with different permissions according to methods, user names and group names. Like other common access control systems, three users groups were define: unknown user, normal user and administrator user.

Screenshots taken during the experiments (see Fig. 10) show that the proposed access control mechanism work as expected. Services for non authorized users could be public (accessible for everybody) or could be some specific service for non authorized, like guest services that are not going to be accessible by authorized users. Services for authorized users could be exclusive for the user (User_name services), exclusive for the user group (Group_name services) or accessible services for authenticated users.

Fig. 10. Screenshots of the available services for a non authorized user / authorized user / administrator taken during the experiment

Table I shows which packet types where an overhead is caused by the access control mechanism proposed in this paper. Depending on ticket size, the overhead (Ticket size) is in normal cases only 6-10 bytes.

	CoAP		CoAP+AA	
	request	response	request	response
GET	N	N	N+T	N
POST	N	N	N+T	N
PUT	N	N	N+T	N
DELETE	N	N	N+T	N
OBSERVE	N	N	N+T	N
ACK	N	N	N	N
RST	N	N	N	N
.well-known/core	N	N	N+T	N
.well-known/auth?login	-	-	I	N+T
.well-known/auth?logout	-	-	O	N

N: Normal size

T: Ticket size

I: Login request size ($I > S$)

O: Logout request size ($O > S+T$)

TABLE I
NORMAL COAP MESSAGE SIZE VS COAP+AAA

VII. FUTURE WORK

The use of a more Kerberos style access control mechanism would be beneficial, therefore the proposed framework will be improved with distributed mechanisms like Kerberos for service access control. The use of RADIUS for 6LoWPAN will also be investigated, i.e. to adopt the same type of mechanisms used by for example WiFi to 6LoWPAN to further increase a networks's security.

The use of a two-way ChallengeResponse mechanism such as CHAP for authenticating a client would improve the security even more since no password, or even a hash of the password, would need to be transmitted. This would never compromise a client's credentials, even in the case when a service has been compromised. By also enabling real-time monitoring of access attempts to services, it is possible to detect when a certain IP address or user name is trying to gain access with incorrect credentials. This intrusion detection system could then be used to notify network administrators about suspicious activity and even blacklist an IP address in for example a firewall. The use of a firewall on each sensor node would of course also be beneficial since rules can be dynamically adjusted to provide an extra security layer by e.g. only allowing certain IP address or address ranges certain access to services.

VIII. CONCLUSION

The use of the CoAP protocol has increased in popularity during the last years, especially over low-bandwidth links such as 6LoWPAN over IEEE 802.15.4. To secure the type of communication, there is the possibility to use IPsec and/or (D)TLS. These two protocols are able to protect the communication channel against some sorts of attacks, but regarding access control both IPsec and DTLS are not sufficient. That is, controlling the access to services by IP and/or session level only does not enable sufficient fine-grain access control. A user on one IP address might have full read and write access to one service, but only read access on another. By using existing methods there is no good way on achieving this. For

that reason the proposed framework suggests a new CoAP Option to be used. This framework proposes uses a type of Ticket as well as defines packet formats to add the possibility to use standard protocol for authentication such as Kerberos and RADIUS. With these definitions this framework's CoAP extension enables a fine grain access control to CoAP-based servers and services.

In the performed experiments, a modified version of the CoAP C-library libcoap [19] was running as a server and a customized version of the Firefox plugin Copper [20] as a client, demonstrating that the proposed CoAP extension works and that fine-grain access control is able to know which IP address, user, services and method that are involved in a request, and send the correct response depending on the client's permissions.

The framework's access control mechanisms have been defined and successfully tested. In the next step, this concept will be converted into a more distributed mechanism like Kerberos, and to test the performance impact in terms of memory usage, power-consumption and packet overhead. Plans are also to add full support using decentralized AAA servers.

ACKNOWLEDGMENT

The authors would like to express their gratitude towards our partners within the Arrowhead project, the European commission and Artemis for funding.

REFERENCES

- [1] S. Karnouskos and A. Colombo, "Architecting the next generation of service-based scada/dcs system of systems," in *IECON 2011 - 37th Annual Conference on IEEE Industrial Electronics Society*, Nov 2011, pp. 359–364.
- [2] G. Candido, F. Jammes, J. de Oliveira, and A. Colombo, "Soa at device level in the industrial domain: Assessment of opc ua and dpws specifications," in *Industrial Informatics (INDIN), 2010 8th IEEE International Conference on*, July 2010, pp. 598–603.
- [3] S. Frankel and S. Krishnani, "IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap," Internet Engineering Task Force (IETF), RFC Editor, RFC 6071, February 2011. [Online]. Available: <http://tools.ietf.org/rfc/rfc6071.txt>
- [4] P. Kasinathan, C. Pastrone, M. Spirito, and M. Vinkovits, "Denial-of-service detection in 6lowpan based internet of things," in *Wireless and Mobile Computing, Networking and Communications (WiMob), 2013 IEEE 9th International Conference on*, Oct 2013, pp. 600–607.
- [5] E. Rescorla and N. Modadugu, "Datagram Transport Layer Security," Network Working Group, RFC Editor, RFC 4347, April 2006. [Online]. Available: <https://tools.ietf.org/rfc/rfc4347.txt>
- [6] Z. Shelby, K. Hartke, and C. Bormann, "Constrained application protocol (coap) draft-ietf-core-coap-18," Tech. Rep., June 2013. [Online]. Available: <http://tools.ietf.org/search/draft-ietf-core-coap-18>
- [7] L. Zhu, K. Jaganathan, and S. Hartman, "The Kerberos Version 5 Generic Security Service Application Program Interface (GSS-API) Mechanism: Version 2," Network Working Group, RFC Editor, RFC 4121, July 2005. [Online]. Available: <http://tools.ietf.org/rfc/rfc4121.txt>
- [8] Y. Rekhter and T. Li, "Remote Authentication Dial In User Service (RADIUS)," Internet Requests for Comments, RFC Editor, RFC 1654, July 1995. [Online]. Available: <http://www.rfc-editor.org/rfc/rfc1654.txt>
- [9] "Monitoring and control: Today's market and its evolution till 2020," 2008. [Online]. Available: ftp://ftp.cordis.europa.eu/pub/fp7/ict/docs/necs/20081009-smart-2_en.pdf
- [10] L. Gide, T. Koljonen, J. Lohstroh, A. ten Berg, and A. Foster, "Artemis strategic research agenda," Artemisis, June 2011.
- [11] D. Kushner, "The real story of stuxnet," February 2013. [Online]. Available: <http://spectrum.ieee.org/telecom/security/the-real-story-of-stuxnet/>

- [12] M. Nottingham and E. Hammer-Lahav, "" defining well-known uniform resource identifiers (uris)," Internet Engineering Task Force (IETF), RFC Editor, RFC 5785, April 2010. [Online]. Available: <https://tools.ietf.org/rfc/rfc5785.txt>
- [13] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," Network Working Group, RFC Editor, RFC 5280, May 2008. [Online]. Available: <http://tools.ietf.org/rfc/rfc5280.txt>
- [14] R. Rivest, "The MD5 Message-Digest Algorithm," Network Working Group, RFC Editor, RFC 1321, April 1992. [Online]. Available: <http://www.ietf.org/rfc/rfc1321.txt>
- [15] D. Eastlake and T. Hansen, "The MD5 Message-Digest Algorithm," Network Working Group, RFC Editor, RFC 4634, July 2006. [Online]. Available: <https://tools.ietf.org/rfc/rfc4634.txt>
- [16] S. Josefsson, "PKCS 5: Password-Based Key Derivation Function 2 (PBKDF2) Test Vectors," Internet Engineering Task Force (IETF), RFC Editor, RFC 6070, January 2011. [Online]. Available: <https://tools.ietf.org/rfc/rfc6070.txt>
- [17] E. V. Fajardo, J. Arkko, J. Loughney, and E. G. Zorn, "Diameter Base Protocol," Internet Engineering Task Force (IETF), RFC Editor, RFC 6733, October 2012. [Online]. Available: <http://tools.ietf.org/rfc/rfc6733.txt>
- [18] S. Raza, S. Duquennoy, J. Höglund, U. Roedig, and T. Voigt, "Secure Communication for the Internet of Things - A Comparison of Link-Layer Security and IPsec for 6LoWPAN," *Security and Communication Networks*, Wiley, Jan. 2012.
- [19] O. Bergmann. (2014, February) libcoap 4.1.1: C-implementation of coap. [Online]. Available: <http://sourceforge.net/projects/libcoap/>
- [20] M. Kovatsch. (2013, November) Copper (cu) v0.18.2, firefox add-on. [Online]. Available: <https://github.com/mkovatsc/Copper>

BIOGRAPHY

Pablo Puñal Pereira is a Ph.D. student in Mobile Internet of Things area at Luleå University of Technology in Sweden. He has received his 5 years degree in Physics in 2008 at the University of Santiago de Compostela, Spain. Also, he has a 5 years degree and a M.Sc. in Electronic Engineering obtained in 2011 at the University of Valencia, Spain. His main research areas involve security, authentication and services for low power wireless networks.

Dr. Jens Eliasson is a researcher in the field of Industrial Internet of Things and received his M.Sc. in Computer Engineering 2003, and Ph.D. in Industrial electronics 2008 and Associate prof. degree in 2014 at Luleå University of Technology in Sweden. He is currently working with Service Oriented Architecture (SOA) for monitoring and control for Internet of Things networks for industrial applications.

Prof. Delsing received the M.Sc. in Engineering Physics at Lund Institute of Technology, Sweden 1982. In 1988 he received the PhD. degree in Electrical Measurement at the Lund University. In 1994 he got the Docent degree (Associate prof) in Heat and Power Engineering. Early 1995 he was appointed full professor in Industrial Electronics at Luleå University of Technology where he currently is working as the scientific head of EISLAB, <http://www.ltu.se/eislab>.