A lightweight mutual authentication protocol for the IoT

Mohamed Tahar HAMMI[1], Erwan LIVOLANT[2] Patrick BELLOT[1], Ahmed SERHROUCHNI[1], and Pascale MINET[3]

[1] LTCI, CNRS, Télécom ParisTech, Université Paris-Saclay, 75013, Paris, France
name@telecom-paristech.fr
[2] AFNet, 30 rue de Miromesnil 75589 Paris, France,
firstname.lastname@afnet.fr
[3] Inria-Paris, EVA team, 2 rue Simone Iff, 75589 Paris Cedex 12, France,
firstname.lastname@inria.fr

Abstract. The Internet of Things enables the interconnection of smart physical and virtual objects, managed by highly developed technologies. WSN, is an essential part of this paradigm. The WSN uses smart, autonomous and usually limited capacity devices in order to sense and monitor industrial environments. However, if no authentication mechanism is deployed, this system can be accessible, used and controlled by non-authorized users. In this paper, we propose a robust WSN mutual authentication protocol. A real implementation of the protocol was realized on *OCARI*, one of the most interesting Wireless Sensor Network technologies. All nodes wanting to access the network should be authenticated at the *MAC* sub-layer of *OCARI*. This protocol is especially designed to be implemented on devices with low storage and computing capacities.

Keywords: Security, Mutual authentication, WSN, IoT, OCARI, MAC Sub-layer, OTP, Industrial Environment.

1 Introduction

According to [11], more than 50 billions of devices will be connected in 2020. This huge infrastructure of devices, which is managed by highly developed technologies, is called *Internet of Things (IoT)*. The latter provides advanced services, and brings economical and societal benefits. This is the reason why thousands of workers and researchers of both industry and scientific community are interested in this area.

Wireless Sensor Network (WSN), is a part of the *IoT* domain. A *WSN* is a network composed of clusters of devices that are equipped with (1) sensors to gather data about the environmental conditions, and/or (2) actuators to interact with the real world. Each cluster is managed by a specific device called *Personal Area Network Coordinator (CPAN)*. Devices are generally characterized by the use of a small computation and memory capacity, low bit rate, low

power consumption, and small packet size. The data produced by each device is transmitted via multiple hops to the *CPAN*, which can use them, or forward them to another network. Usually *WSN* technologies are based on IEEE 802.15.4 physical layer (*PHY*) [9] that provides a good foundation for building ad-hoc mesh networks.

Optimization of Communication for Ad hoc Reliable Industrial networks (OCARI) [5] is a promising WSN. It is characterized by its optimized energy consumption, its time-constrained communication at the *MAC* sub-layer, and its support of pedestrian mobility [1]. However, it needs to be secured against the different threats, especially those that concern confidentiality, data integrity, and entities authentication.

In order to secure *OCARI* specification, our work aims to create a robust security protocol, which ensures a mutual authentication between the device and the *CPAN* at the *MAC* sub-layer, to protect the system against malicious intruders. The proposed protocol also provides a secure algorithm for the exchange of symmetric keys (used to ensure the data integrity). A real implementation with C language, deployed on the OCARI platform, were realized in this work. In this paper, the authentication and data integrity services are our primary concern. We do not consider the confidentiality service.

The rest of this paper is organized as follows. Section 2 presents the related *WSN* and *IoT* technologies and their authentication mechanisms. Section 3 describes our proposed approach and its implementation. Section 4 details a real tests that we have made. Then we provide an evaluation of our authentication protocol. Finally, our conclusions and future work are drawn in section 5.

2 Related work

IoT and networks in general, represent the working environment of hackers. Everyday, companies and individuals are victims of different kinds of attacks: Denial/Distributed Denial of Service (Dos/DDos) attacks, usurpation of identity, intrusions, data theft, etc. In return, several researches have been realized in order to secure and protect the Information Technology (IT) systems.

In [8], we proposed an authentication protocol based on pre-shared keys. It provides only the authentication of a device during its association to a cluster. Although this solution is lightweight and fast, the authentication of the CPAN is missing in this work. In addition it does not use a good mechanism for the generated keys exchange (this keys are required for the data authentication once the association is realized).

An interesting work described in [3] intends to secure the IoT devices. They propose an authentication mechanism based on shared key between constrained (Cd) and unconstrained (Ud) devices. Both sides use the same security policy, and no gateway is required. For an easier understanding, authors give an example using an IPsec-based security association (see Fig 1). First, the concerned entities agree on the security policy. Then, exchange the keying material. After, authenticate each other. And finally, create a secure channel. This mechanism is

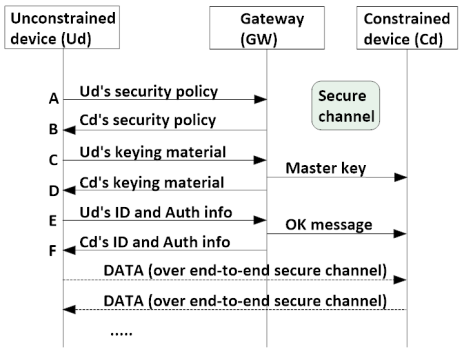

Fig. 1. Association and secure channel establishment solution

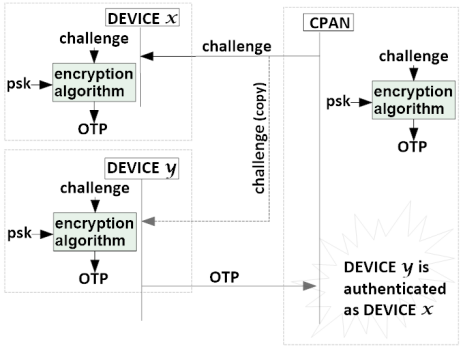

Fig. 2. An asynchronous authentication operation

based on IPsec, which is known by its robustness. However, it has some weaknesses: (1) In C (Fig 1), the GW, receives the Ud's keying material, and without authenticating Ud, generates and sends a master key to the constrained device. In this case if Ud is malicious, it can send a big number of C messages to the gateway, which, therefore, sends master keys to the constrained device. Knowing that the reception of messages consumes a lot of energy, as a result, this increases the energy consumption of the constrained device. (2) In D, without authenticating the gateway, Ud generates it's own master key. Thus, if a malicious GW generates a Dos/DDos attack (by sending a lot of data about the keying material, or a lot of D messages), this can stop the operation of Ud.

Authors in [4], propose an authentication mechanism for the IEEE 802.1x technologies, based on Extensible Authentication Protocol (EAP). In order to ensure a secure communication between two entities, first, they exchange their identities (without any proof). Then, an authentication server (Remote Authentication Dial-In User Service server) is involved to check the authenticity of the entities, using algorithms and mechanisms as MD5 or TLS protocol. This solution is flexible, based on standardized algorithms, and can be deployed on different systems. However it can not be deployed without a trusted third party (authentication server). Furthermore, it requires the exchange of a very big number of messages (10 messages). Thus, the execution time and the energy consumption of this approach is very high.

One can note that works seen above does not sufficiently address the problem of the energy and time constraints, and can have some weaknesses. In this paper, we propose an approach that provides an energy efficient and optimal mutual authentication method, as well as a key exchange algorithm, designed for WSN systems.

3 Proposed approach

We implemented our authentication mechanism in the MAC sub-layer, in order to provide more transparency and interoperability in the highest layers (network and application).

3.1 Algorithms

To ensure a mutual authentication in the MAC sub-layer association step, we designed a method based on an **asynchronous** One Time Password (OTP) using a Challenge/Response mechanism. We opted for the asynchronous mode, because generally WSNs does not support an *absolute time* of sufficient precision, which can be exploited in the synchronous mode. In addition, this mode compared to the **synchronous** one (see RFC4226 [10]), does not require any prior approval between the communicating entities, which represents a great advantage, and this allows more flexibility for the system. The OTP by definition is a password, valid for only one transaction, used for proving the identity of an entity. In other words, even if it is transmitted without any encryption, a malicious user cannot exploit it to authenticate itself. Fig 2 shows a possible architecture using an asynchronous authentication operation in the association step.

All the devices of the same cluster have the same secret pre-shared key psk. If a device x wants to join the cluster, it computes an OTP using a received random number ($challenge$) and the psk), then sends it to the CPAN. However, any device y which can catch the transmitted $challenge$ can generate the OTP (because it has the same psk), and use it to authenticate itself on behalf of device x. In other words, y steals the identity of x and gets all the authentication/encryption keys, that normally should be secret and shared only between x and the CPAN. Therefore, the data integrity and confidentiality are no longer ensured.

3.2 Preparation of nodes

Our protocol allows a mutual strong authentication, and solves the problem of the internal identity usurpation due to the "personalization" of the keys. This operation is described in the Fig 3. The trusted authority, which is generally the provider, should setup in out-of-band channel, what we call "key_{mother}" into the CPAN, and derives from it a personalized "$key_{daughter}$" attributed to each legitimate device belonging to the same cluster. The derivation of this keys is based on the function $f(key_{mother},\ UI)$, where UI (Unique Identifier) is the 8 bytes device's IEEE address. This function is defined as below:

$$\begin{cases} key_{daughter} = f(key_{mother},\ UI) \\ f(key_{mother},\ UI) = hash\_func(key_{mother},\ UI) \end{cases} \tag{1}$$

Where: $hash\_func$ is an irreversible function that generates a strong key, and which protects the key_{mother} against deductive attacks. Once the $key_{daughter}$ is created and set into the device, the latter becomes able to be associated to the cluster.

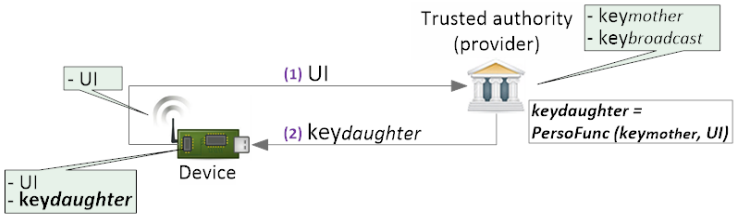

Fig. 3. The personalization of keys

3.3 Association step

Fig 4 illustrates our mutual authentication protocol in the MAC sub-layer association process. During the association step, the personalized $key_{daughter}$ of a device D can be generated only by the CPAN that has the appropriate key_{mother}. Thus this $key_{daughter}$ is known only by D and by the CPAN, which ensures the protection of the $key_{daughter}$. To start the association operation, the device sends an "association request" message to the CPAN, if needed the request passes through one or several device(s), called relay(s), to reach the intended destination. Receiving the request, the CPAN checks if the UI of the device is blacklisted or not. (1) If it is the case, the association request is directly rejected and no processing is done. This method prevents the system from reserving uselessly the memory and doing additional processing, thus this protects the CPAN from some kind of DoS attacks. (2) Otherwise, it answers by an "authentication request" containing a *challenge*. With the received *challenge*, its own $key_{daughter}$ and using an *encryption algorithm*, the device computes "$otp1$" and sends it through an "authentication response" message. For the *encryption algorithm* we opted for the *HMAC-Based One-Time Password Algorithm* described in the RFC4226 [10] which is proposed for the **synchronous** OTP mode, and which basically uses an increasing counter value. After modifications, we compute $otp1$ using the function below (function 2):

$$\begin{cases} HOTP(key,\ challenge) = \\ Truncate(HMAC - SHA256(key,\ challenge)) \end{cases} \tag{2}$$

Receiving the device's response, the CPAN generates the appropriate $key_{daughter}$ using the same personalization function (described above in (subsection 1), and computes "$otp1\prime$". Then compares the two otps, if they match then the device is authenticated, otherwise the association operation fails. If the same device is rejected consecutively MAX_ASSOC_REQ times, then it is blacklisted. The fact that there is only the legitimate device that can have the $key_{daughter}$ which is used to compute $otp1$, represents a proof of for its identity. Once the latter is authenticated, the CPAN generates an authentication key called "key_{auth}" using the standard Pseudo Random Function (PRF) defined in the SSL/TLS specification [6]. This key will be used to securely exchange the broadcast key "$key_{broadcast}$", and then to authenticate the exchanged data after the association

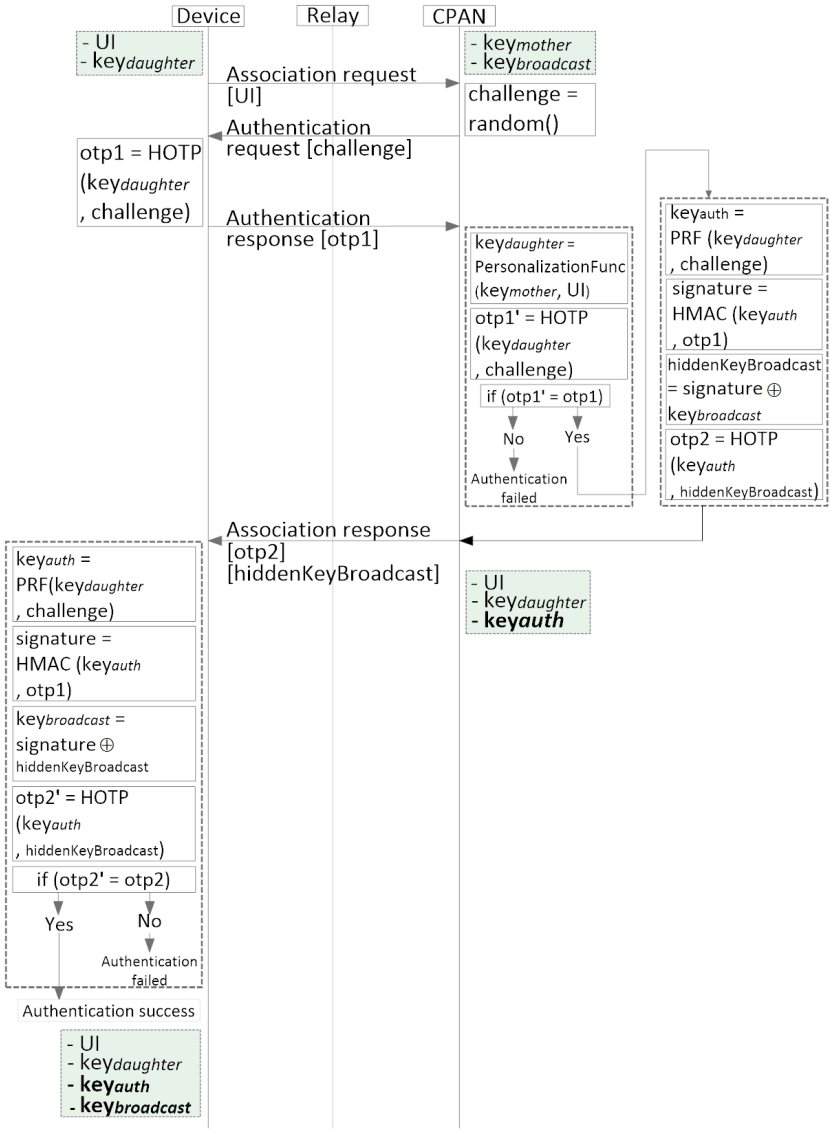

Fig. 4. Our proposed mutual authentication protocol

step in the unicast mode. The $key_{broadcast}$ is used to ensure the integrity of the broadcasted messages (after the association step). To share this key with the authenticated device, the CPAN should hide it into a "$hiddenKeyBroadcast$" value using the function below (function 3):

$$\begin{cases} hiddenKeyBroadcast = signature \oplus key_{broadcast} \\ where: signature = HMAC - SHA256(key_{auth}, otp1) \end{cases} \quad (3)$$

We created this function in order to securely share the $key_{broadcast}$ (Fig 5 shows the visble information for the different kind of users). (1) If an external attacker intercepts all the exchanged information ($challenge$, $otp1$, $hiddenKeyBroadcast$, and $otp2$ (explained below)), it can not get any secret information (keys), because it does not have the couple ($key_{daughter}$, $key_{broadcast}$) nor (key_{auth}, $key_{broadcast}$). (2) For an internal attacker which has the $key_{broadcast}$ in addition to all the exchanged information, it cannot also get the keys of other devices. That is to say, when an internal attacker attempts to get the key_{auth} of another device, it computes the xor ($\oplus$) between the $key_{broadcast}$ and the $hiddenKeyBroadcast$ in order to obtain the $signature$, and because the latter is generated by an irreversible function (HMAC), even using $otp1$, the attacker cannot get the key_{auth}. $otp2$ is computed by the CPAN for hitting two targets with one shot. Firstly to ensure the integrity of the $hiddenKeyBroadcast$, and secondly to authenticate itself. To be generated, $otp2$ needs a secret (key), and a unique challenge. For this reason the CPAN uses key_{auth} as a secret, and exploits the $hiddenKeyBroadcast$ as challenge. The latter is unique, because it is based on a unique signature, that is based on unique otp ($otp1$). Then $otp2$ is sent accompanied by the $hiddenKeyBroadcast$ through an "association response" message. Finally, when the device receives the message, it computes also key_{auth} and $signature$ using the same inputs applied by the CPAN. Then to retrieve the $key_{broadcast}$ it computes the xor between $signature$ and $hiddenKeyBroadcast$ (see following function 4):

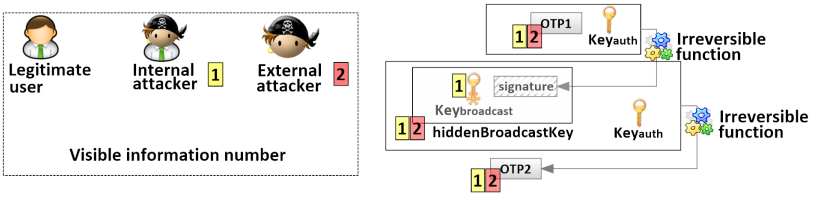

Fig. 5. HiddenBroadCastKey mechanism

$$key_{broadcast} = signature \oplus hiddenKeyBroadcast \tag{4}$$

The device gets a $key_{broadcast}$ which needs to be verified (check for its integrity). That is why it computes $otp2\prime$ based on the received $hiddenKeyBroadcast$ and key_{auth}, then the device compares the two otps, if they match, then this means that the $hiddenKeyBroadcast$ is correct, thus the $key_{broadcast}$ is correct and the CPAN is authenticated. Otherwise if the retrieved $otp2$ or the $hiddenKeyBroadcast$, or both of them are wrong or modified during their transmission, then $otp2$ and $otp2\prime$ will not match, hence the $key_{broadcast}$ is not accepted, the CPAN is not be authenticated, and the association operation stops. The fact that the device receives from the CPAN a correct $otp2$, validates the identity of the CPAN. Because $otp2$ is computed by key_{auth} which is derived from

$key_{daughter}$. Accordingly, the protocol ensures a mutual authentication and a secure exchange of keys. Once the DEVICE is associated to the network, it will use the obtained keys (key_{auth} and $key_{broadcast}$) for the authentication of the exchanged messages under an authenticated channel. In fact, exchanged packets in the authenticated channel should be signed. Which means that each packet, which is made of a header and a body, should add a signature in order to ensure its integrity. The signature is an HMAC of 16 bytes of the header and the body computed using the key_{auth} (unicast mode) or the $key_{broadcast}$ (broadcast mode).

4 Test and evaluation

For testing our authentication mechanism, we implemented our protocol in the OCARI stack software [1]. OCARI technology is based on the IEEE 802.15.4 physical layer, which is adapted to harsh environment such as power plants and factories. This layer ensures a good signal transmission, that is resilient to radio interferences. Unlike the IEEE 802.15.4 physical layer, the IEEE.802.15.4 MAC layer was replaced by "MaCARI", that is designed taking into consideration two different factors that are "determinism" and "energy optimization". As explained in [1], a deterministic MAC layer should guarantee an access to the medium for each node, every certain period of time. While an energy-efficient MAC layer has to make all the nodes sleep as much as possible. MaCARI uses different access methods to the medium. It uses CSMA/CA for control messages, combined with TDMA for data messages.

Our code is developed with C language and the hardware used is Dresden Elektronik deRFsam3 23M10-R3. It has 48 kb of RAM, 256 kb of ROM and a Cortex M-3 Processor. There is no specific hardware for the authentication mechanism. For our experiment, we created an architecture composed of 2 devices and one CPAN. First, we personalized the devices by flashing into them their associated keys ($key_{daughter}$) as explained in Section 3.2. Using a Zolertia z1 sniffer (hardware) and Wireshark [7], we can follow the association and the authenticated channel establishment operations. Fig 6 shows the exchanged frames during the association step. The two "Unknown Command" represents respectively: the "Authentication request message" and the "Authentication response message". For OCARI, at the beginning, devices at 1 hop from the CPAN sends directly an association request to it. Then device at 2 hops sends a request to the CPAN by means of the authenticated ones, that play the role of router. Then we measured the delays of the association operation with (auth) and without (none) the authentication procedure, and compared this results with studies and evaluations of a 1 hop device analyze, realized on an implementation of the Zigbee protocol (WSN technology) [2], using a similar hardware equipment. As shown in Fig 7, the average delay of the association operation is increased from 0,5243 ms without authentication to 34,45 ms with authentication for the DEVICE at 1 hop from the CPAN. The association time needed by the DEVICE at 2 hops from the CPAN is also increased from 34,5076 ms without authentication

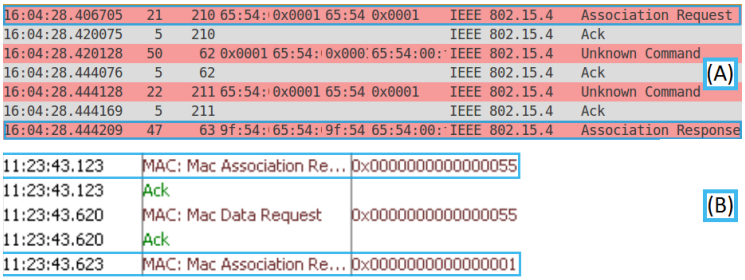

Fig. 6. A Wireshark capture of (A) OCARI and (B) Zigbee frames during a node association step

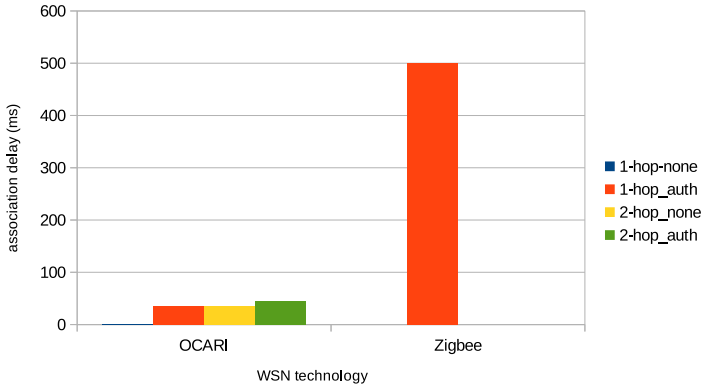

Fig. 7. Association delays comparison with and without authentication

to 45,876ms with authentication. This increase is mainly due to the exchange of additional messages of the authentication protocol. Indeed, the number of messages needed by the 2 hop DEVICE for its association without authentication is 4 which is the same as the number of messages for the 1 hop DEVICE association with authentication. The difference between both is only 0,0576ms (34,5076 34,45). Hence, the increase in time caused by the authentication is due to these additional messages and in a very lesser extent to an increase in processing time.

In a similar way as in OCARI, the execution time of the secured Zigbee association can be computed by the subtraction between the Association response message and the Association resquest message timestamps. The delay of 1 hop with authentication association is equal to 500 ms (11h:23m:43.623s - 11h:23m:43.123s). It is true that this big difference between association delays of OCARI and Zigbee is due also to the nature of the two technologies, but still, the used security protocol plays a main role.

With the real implementation, we proved that our solution is robust, ensures a good mutual authentication and a secure key exchange mechanism.

5 Conclusion and future work

Our approach is based on a lightweight, robust, and energy efficient mechanism that allows to solve the problem of the WSN mutual authentication at the MAC sub-layer. This solution provides a protection against "replay attacks", because the exchanged OTPs are based on random numbers, therefore, they are valid only for one transaction. Using the blacklisting mechanism we can secure our systems against "some DoS" attacks. Finally it is flexible and does not decrease the scalability of the system, and can be deployed in different WSNs technologies, while keeping the same level of robustness.

In our future work we aim to ensure the confidentiality of the transmitted messages exchanged after the MAC sub-layer association and authentication procedure. And thus we will have a secure system which ensures the "Confidentiality", "Integrity, and "Authentication" services.

References

1. Agha, K.A., Bertin, M.H., Dang, T., Guitton, A., Minet, P., Val, T., Viollet, J.B.: Which Wireless Technology for Industrial Wireless Sensor Networks? The Development of OCARI Technology. vol. 56, pp. 4266–4278 (Oct 2009)
2. Atmel: Zigbee pro pack and analysis with sniffer. In: Application note (sep 2013)
3. Bonetto, R., Bui, N., Lakkundi, V., Olivereau, A., Serbanati, A., Rossi, M.: Secure communication for smart iot objects: Protocol stacks, use cases and practical examples. In: World of Wireless, Mobile and Multimedia Networks (WoWMoM). pp. 1–7 (June 2012)
4. Chen, J.C., Wang, Y.P.: Extensible authentication protocol (eap) and ieee 802.1x: tutorial and empirical experience. vol. 43, pp. supl–26–supl–32 (Dec 2005)
5. Dang, T., Devic, C.: OCARI: Optimization of communication for Ad hoc reliable industrial networks. In: Industrial Informatics, 2008. INDIN 2008. 6th IEEE International Conference on. pp. 688–693. IEEE (2008)
6. Dierks, T.: The transport layer security (TLS) protocol version 1.2 (2008)
7. Foundation, T.W.: Wireshark 2.0.3 and 1.12.11 Released (April 22, 2016), https://www.wireshark.org/news/20160422.html, https://www.wireshark.org/news/20160422.html accessed online on 2016-06-28
8. Hammi, M.T., Livolant, E., Bellot, P., Serhrouchni, A., Minet, P.: Mac sub-layer node authentication in ocari. In: 2016 International Conference on Performance Evaluation and Modeling in Wired and Wireless Networks (PEMWN). pp. 1–6 (Nov 2016)
9. IEEE: IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs). IEEE (September 2011)
10. M'Raihi, D., Bellare, M., Hoornaert, F., Naccache, D., Ranen, O.: HOTP: An HMAC-based one-time password algorithm. IETF (December 2005)
11. statista.com: online statistics portal, and one of the world's most successful statistics databases (2016)