A Lightweight IoT Security Protocol

Mohamed Hammi, Erwan Livolant, Patrick Bellot, Ahmed Serhrouchni, Pascale Minet

A Lightweight IoT Security Protocol

Mohamed Tahar Hammi*, Erwan Livolant†, Patrick Bellot*, Ahmed Serhrouchni*, Pascale Minet‡
* LTCI, Télécom ParisTech, Université Paris-Saclay, 75013, Paris, France
*{hammi,bellot,serhrouchni}@telecom-paristech.fr
†AFNeT, Boost-Conseil, 75008 Paris, France
†erwan.livolant@boost-conseil.com
‡Inria-Paris, EVA team, 2 rue Simone Iff, 75589 Paris Cedex 12, France
‡pascale.minet@inria.fr

***Abstract*—The IoT is a technology that enables the interconnection of smart physical and virtual objects and provides advanced services. Objects or things are generally constrained devices which are limited by their energy, computing and storage capacity. A *Wireless Sensor Networks (WSN)* is a network composed of devices managed by a CPAN (Personal Area Network Coordinator). The network is used in order to gather and process data of a given environment. It is characterized by their low bit rate and low power consumption, and it uses small size packet in their transmissions. In order to protect the WSN, a mutual authentication between devices is required during the association of a new device. The exchanged data should be authenticated and encrypted. In this work we propose a robust, lightweight and energy-efficient security protocol for the WSN systems. The real tests we made and a performance evaluation of our security protocol are provided.**

***Index Terms*—Security, Authenticated encryption, Mutual authentication, WSN, IoT, Industrial Environment, Scyther, OCARI.**

I. Introduction

With the IoT, today, virtual and smart physical things are able to communicate with each other, without the human intervention. This technology attracts different fields, because of its economical and societal benefits. Industry is in the top list. In fact, industrial wireless sensor networks represent a sub-domain of the IoT which concerns limited capacity devices used to gather data and manage various environments. Due to the nature of these devices, security represents a big issue for researchers and developers. In this paper, we present a robust and a lightweight securing protocol that ensures a mutual authentication and secures transmissions between devices. This protocol has been implemented on the OCARI platfrom which is a promising and energy efficient industrial wireless sensor network.

In 2014, according to *The New York Times*, Russian hackers got access to the state department's unclassified system and stole the archived e-mails including the president's ones. Even worst, a few years ago in Australia, a certain *Vitek Boden*, for revenge reasons, attacked the SCADA (Supervisory Control and Data Acquisition) system of the *Maroochy Shire Council* which was in charge of the waste management. As a result, millions of liters of raw sewage were redirected to a park and an hotel located around the company. Thus, an ecosystem was destroyed in short time [1]. In the UK, a provider of telecommunications services, was the victim of an attack in October 2015. Hackers managed to get many customers records, that contain important informations like logins and passwords, secret codes, confidential and personal data, etc. Therefore the company has lost a lot of money and customers.

The unauthorized access, the identity usurpation, and the steal and/or modification of stored and/or exchanged data represent a serious danger for our information systems. Researchers and developers try to find and create new solutions to enhance the security and the robustness of such systems. The issue is more complex if the system is an architecture for the Internet of Things (IoT). The IoT is a technology that enables the interconnection of smart physical and virtual objects and provides advanced services. Objects or things are generally constrained devices, which are limited by their energy, computing and storage capacity.

The IoT covers a lot of areas such as *Smart Cities*, *M2M (Machine to Machine) systems*, *Body Area Networks (BAN)*, and *Wireless Sensor Networks (WSN)*. The WSN is a network composed of devices managed by a CPAN (Personal Area Network Coordinator). The network is used in order to gather and process data of a given environment. Usually, the devices (except the CPAN) are limited in terms of computation and memory capacity. They are characterized by their low bit rate and low power consumption, and they use small size packet in their transmissions. The data produced by each device is transmitted via multiple hops to the CPAN, which can use them, or forward them to another network. The most known WSN technologies are based on the IEEE 802.15.4 physical layer (PHY) [9]. It is resilient to radio interferences and provides a good foundation for building ad-hoc mesh networks.

In order to protect the WSN, a mutual authentication between devices is required during the association of a new device. The exchanged data should be authenticated and encrypted. In this work we propose a robust, lightweight and energy-efficient security protocol for the WSN systems. This paper is organized as follows. Section II presents different researches realized for securing the IoT. Section III explains our proposed approach and its implementation. The real tests we made and a performance evaluation of our security protocol are provided in section IV. Finally a conclusion and our future works are given in section V.

II. Related work

In [7], we proposed an authentication protocol for securing the IoT system OCARI, an industrial WSN for constrained environment. It was based on pre-shared keys. It provided a mutual authentication and a good mechanism for the derived key exchange during the association step. Although this solution is lightweight robust and fast protocol, the confidentiality of the transmitted data is missing. In addition using HMAC [11] for signing packets can be expensive in terms of computing and execution time.

Authors in [13] propose a security protocol called HIP (Host Identity Protocol). Their study focuses mainly on the security of the constrained devices over LoWPAN (Low power Wireless Personal Area Networks). HIP is based on the asymmetric key cryptography. They propose to add a central authority for managing and controlling each IoT domain. During a new association, both devices and the central authority should be mutually authenticated using the asymmetric cryptography. Once the two communicating entities are authenticated, session symmetric keys are shared and an encrypted communication can start. For managing and updating keys, they use another protocol called MIKEY (Multimedia Internet KEYing). Their solution ensures a good authentication and data protection mechanism. However generating and providing new keys for each association step consumes a lot of time and energy. In Section III, we will describe our system that does not require a generation of new symmetric keys while keeping the system safe against the cryptanalysis and replay attacks.

Researchers in [10] propose a secure architecture based on the DTLS (Datagram Transport Layer Security) protocol for IoT. It was designed to work over LoWPANs. It provides a mutual authentication and symmetric key exchange for creating a symmetric secure channel). It uses x509 certificates [8] and RSA (Rivest Shamir Adleman) algorithm [15]. The authentication requires a trustful third parties. Although this solution is very robust and ensures a strong authentication mechanism, it is not optimized. First, the use of RSA algorithm and the exchange of a large number of messages (6 messages are mandatory for the DTLS-Handshake) is resource-intensive computing and consumes a lot of energy. Secondly, the size of x509 certificates is not adapted to the constrained devices that have a small capacity of memorization. And thirdly, this solution is time consuming, as shown in [10], the execution time with keys of 1024-bit of the DTLS-handshake takes 3783 ms for the single hop, and 4791 ms for the multi hop. Using 2048-bit keys, it takes 4000 ms for the the single hop and 6627 ms for the multi hop. This represents a large execution time that could not be acceptable for applications with strong latency constraints.

Another interesting work described in [16] proposes a lightweight proposition designed for the WIFI based IoT. In their architecture, all the communications should pass through a gateway. This solution provides a mutual authentication based on the public keys combined with the pre-shared keys. It uses the Elliptic Curve Diffie-Hellman (ECDH) operation to generate a shared key. It will be used to secure transmissions during the data exchange using symmetric secure channel. This solution requires the exchange of only 3 messages. It is lightweight, energy-efficient, and needs a reasonable computing capacity. The only weakness is in the first sent message (A) in Figure 1.

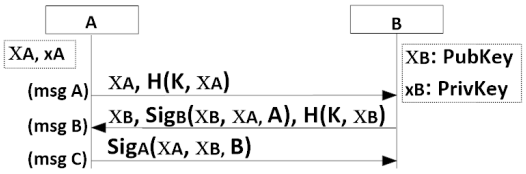

Figure 1: Authentication Procedure for a Wi-Fi based IoT

X_A, X_B and x_A, x_B represent respectively the public and the private keys. K is a pre-shared key between two entities. $H(u, v)$ is the hash of the message u, v. And finally $Sig_A(u, v, w), Sig_B(u, v, w)$ is the signature of the messageu, v, w of the entity with its own public key. The messages labelled (B) and (C) are without any risk. However the authentication with the pre-shared key K in message labelled (A), without any random factor or counter, exposes K to potential cryptanalysis attack.

In the next section, we will explain our proposed approach, which aim is to solve the performance and security problems seen above.

III. Proposed approach

As explained above, the WSN architecture is made of constrained devices (sensors, or actuators) managed by a CPAN which is an unconstrained device. When a new device wants to join the WSN network, first a mutual authentication should be ensured. Then, a symmetric secure channel should be created between the communicating entities in order to protect the exchanged data. The mutual authentication mechanism was realized in the MAC sub-layer, and the authenticated encryption of data is done in the application layer.

A. Chosen algorithms

For the authentication mechanism, we opted for the asynchronous OTP (One Time Password) algorithm. It is adapted to our needs [6]. The OTP is a password that can be used only one time. It is based on pre-shared key and a random challenge in the case of asynchronous OTP. The random challenge protects the authentication against replay attacks and cryptanalysis attacks. In order to ensure a robust and fast authenticated encryption of data, we implemented the AES, also called Rijndael, possibly using GCM (Galois Counter Mode) or CCM (Counter with CBC-MAC). The mode to use is selected in the configuration file. The authenticated encryption ensures the confidentiality and the integrity of the transmitted data in the same time.

We deployed our protocol in the OCARI network (Optimization of Communication for Ad hoc Reliable Industrial networks). OCARI is an energy efficient WSN technology. It

represents an application of the IoT in the industrial environment. OCARI is based on the IEEE 802.15.4 physical layer [9] that allows reliable signal transmission and resistance against the radio interferences in harsh environment (eg. power plants, factories, etc). We previously designed and implemented a new security mechanism for the authentication of the associated devices and the integrity of their exchanged data [6]. However the confidentiality service was still missing. Although this solution is proposed and implemented for OCARI, it can also be deployed for any other WSN.

B. Design of our protocol

For the key management, we created a method called the "personalization". The principle of this method is detailed in Figure 2. For each OCARI network, the devices provider generates a "kit" of secret keys that contains: an initial key key_i and derived onces $key_d(s)$. The kit will be installed, in out of band mode, in the CPAN and the concerned devices. The derived keys are computed from unique identifier (UI) of each device and key_i using the "PersoFunc()" function (see equation 1). It is an irreversible function that generates a strong key and protects the key_i against deductive attacks.

$$persoFunc(key_i, UI) = HMAC(key_i, UI) \quad (1)$$

Once the key_d is created and set into the device, the device is able to be associated with the OCARI network.

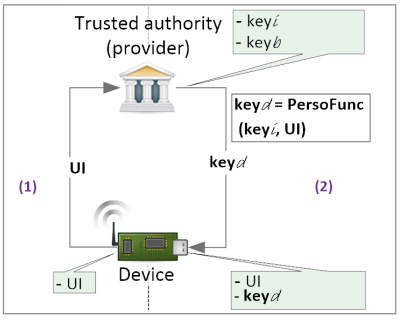

Figure 2: The personalization of devices

The goal of this personalization is to ensure that the communication between a device A and the CPAN cannot be intercepted by a device B belonging to the same OCARI network. In addition, the other advantage is that even if an attacker could get a personalized key of one device, it will not influence the security of the rest of devices belonging to the same OCARI network.

Figure 3 on the following page depicts the association of a device to a cluster. We can summarize this process:

- The device sends an association request to the CPAN. It receives an authentication request that contains a challenge (random number). By means of the encryption algorithm named HOTP [14]), it computes the OTP using its key_d and the challenge. It sends the generated otp_1 to the CPAN as the authentication response.
- The CPAN checks if the joining device is blacklisted or not. Then, it generates the key_d for this device, computes otp'_1, and compares the latter with $otp1$. If the authentication failed and the same device has been rejected consecutively max_assoc_req times, then it is blacklisted. Otherwise the authentication is successful. It generates a symmetric secret key called key_u. This key will be used for the authenticated encryption of the exchanged messages in the unicast mode. It computes otp_2 for the authentication of the CPAN, and hides the key_b. The key_b is the authenticated encryption key in the broadcast mode:

$$key_u = PRF(key_d, challenge)$$
$$signature = HMAC(key_u, otp_1)$$
$$hiddenKeyBroadcast = signature \oplus key_b$$
$$otp_2 = HOTP(key_u, hiddenKeyBroadcast)$$

  where PRF is the Pseudo Random Function defined in [3], used to create the key_u and securely share the key_b. If an external attacker intercepts all the exchanged information, $challenge$, otp_1, $hiddenKeyBroadcast$, and otp_2, it cannot compute any secret information because it does not have the couple (key_d, key_b) nor (key_u, key_b). For an internal attacker which has the key_b in addition to all the exchanged information, it cannot get the keys of other devices. That is to say, when an internal attacker attempts to get the key_u of another device, it computes the xor between the key_b and the $hiddenKeyBroadcast$ in order to obtain the $signature$ and because the latter is generated by an irreversible function, even using otp_1, the attacker cannot get the key_u. otp_2 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, otp_2 needs a secret key and a unique challenge. For this reason the CPAN uses key_u as a secret and exploits the $hiddenKeyBroadcast$ as the challenge. The latter is unique, because it is based on a unique signature, that is based on unique OTP (otp_1). Then otp_2 is sent accompanied by the $hiddenKeyBroadcast$ through an association response.
- Finally, when the device receives the message, it computes also key_u and $signature$ using the same inputs as the CPAN. It retrieves the key_b by xoring $signature$ and $hiddenKeyBroadcast$. The device gets a key_b which needs to be verified by checking for its integrity. That is why it computes otp'_2 based on the received $hiddenKeyBroadcast$ and key_u. Then the device compares the two otps. If they match, this means that the $hiddenKeyBroadcast$ is correct, thus the key_b is correct and the CPAN is authenticated. Otherwise if the retrieved otp_2 or the $hiddenKeyBroadcast$ or both of them are wrong. The may have been modified during their transmission, then otp_2 and otp'_2 will not match. Hence the key_b is not accepted, the CPAN is not authenticated, and the association operation stops.

The secure channel created after the association step uses AES with GCM or CCM modes of operation. In following,

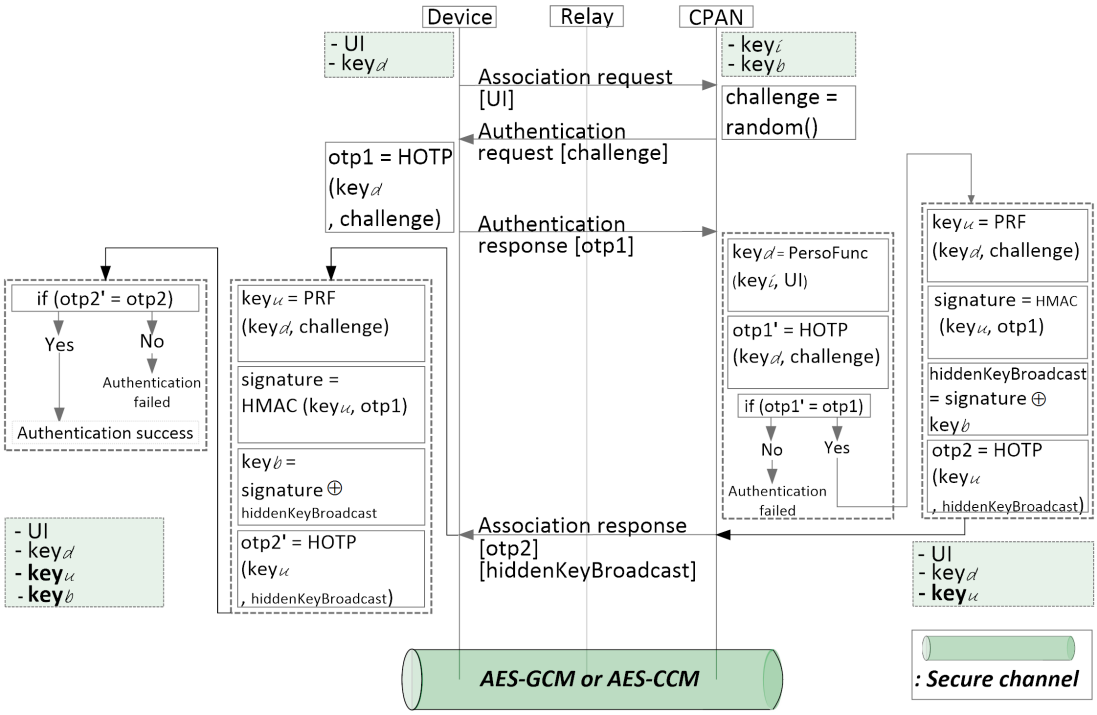

Figure 3: OCARI secured association

we will consider only the AES-GCM, where GCM is a mode of operation for block ciphers that uses universal hashing over a binary Galois field [4].

1) Authenticated encryption: First, the entity that wants to send data, should authenticate and encrypt its data using the generated key_u in the unicast mode, and the key_b in the broadcast mode. This operation requires a plaintext $P_1..P_n$, an additional authenticated data A (A can be any random data, added for strengthen the encryption algorithm) and an initialization vector IV as inputs. The generation of the IV is based on the key_u and a counter value. The latter is used in order to avoid the cryptanalysis attacks. And as a result, we get the ciphertext $C_1..C_n$ and an authenticated tag T.

The authenticated encryption operation is defined by the following equations (2):

$$\begin{cases} H = E(K, 0^{128}) \\ \text{We use } len(IV) \text{ of 96 bits} \\ \quad => Y_0 = IV||0^{31}1 \\ Y_i = Y_{i-1} + 1, for\, i = 1..n \\ C_i = P_i \oplus E(K, Y_i), for\, i = 1, ..n \\ T = MSB_t(GHASH(H, A, C, len(A), len(C)) \\ \quad \oplus E(K, Y_0)) \end{cases} \tag{2}$$

where E is the encryption operation, K is the secret key (K_u or K_b), 0^{128} is a block of 128 bits of 0, H is obtained by encrypting a zero block using K, Y is a counter starting from Y_0 which is the concatenation ($||$) of IV with 31 zeros and one (bits), MSB_t is the most significant bit, and $len(A)$ is the length of A and $GHASH()$ is the hash function of the GCM mode of operation.

In the end of the authenticated encryption operation we usually concatenate the *ciphertext* with the tag *T*. The latter will be used for checking the integrity of the message once the packet is received.

2) Authenticated decryption: We have four input parameters for the authenticated decryption operation: the ciphertext C, the received tag T, the IV and the authenticated additional data A. And as output, we get the plaintext P and a Tag T' for checking the integrity of data. Compared to the encryption operation, the order of the hash step and decrypt step are reversed. The authenticated decryption operation is defined by the following equations:

$$\begin{cases} H = E(K, 0^{128}) \\ \text{We use } len(IV) \text{ of 96 bits} \\ \quad => Y_0 = IV||0^{31}1 \\ T' = MSB_t(GHASH(H, A, C, len(A), \\ \quad len(C)) \oplus E(K, Y_0)) \\ Y_i = Y_{i-1} + 1, for\, i = 1..n \\ P_i = C_i \oplus E(K, Y_i), for\, i = 1, ..n \end{cases} \tag{3}$$

At the end of the authenticated decryption operation, we compare the received T with T'. If they match, the decryption operation is successful. Otherwise the operation fails. For more details about the GCM mode of operation see [5].

IV. Test and evaluation

A. Formal validation

In order to check the robustness and the safety of our protocol, we realized a formal validation using *Scyther* [2].

Which is a tool for the automatic verification of security protocols created by *Cas Cremers* at the university of *Oxford*. In *Scyther* formal language, each protocol is defined by "roles". And each role should be played by an "agent", and described by a sequence of events (send, receive,..etc). In the following, we show the structure of our code:

```
the definition of new types
... etc

protocol  OCARIAuthAndEncProtocol(D,C)
{
function OneTimePassword          ;
function keyGeneration            ;
macro    otp1 = OneTimePassword
              (k(D,C),challenge);
// k(D,C) is the personalized
// symmetric key between D and C
macro    keyAuthAndEnc =
             keyGeneration(k(D,C)
             ,challenge)          ;
macro    otp2 = OneTimePassword
               (keyAuthAndEnc
             ,hiddenKeyBroadcast);
hashfunction signFunc             ;
macro    signature = signFunc
             (keyAuthAndEnc,otp1);
function HiddeBroadCastKey        ;
function RevealBroadCastKey       ;
const    deviceAuthError : String;
const    cpanAuthError   : String;
const    securedPacket            ;
```

```
role D {
the declaration of the local variables
... etc
recv_1(C,D,challenge)             ;
send_2(D,C,otp1)                  ;
recv_3(C,D,(receivedOtp2
           ,hiddenKeyBroadcast));

match(receivedOtp2,otp2)          ;
  // Ok, CPAN auth successful
  macro broadCastKey =
     RevealBroadCastKey(signature
             ,hiddenKeyBroadcast);
  send_4(D,C,
   ,{securedPacket}keyAuthAndEnc);

claim (D, SKR, keyAuthAndEnc)     ;
claim (D, SKR, broadCastKey)      ;
claim (D, Alive)                  ;
claim (D, Weakagree)              ;
claim (D, Niagree)                ;
claim (D, Secret, securedPacket)  ;
}
```

```
role C {
the declaration of the local variables
... etc
send_1(C,D,challenge)             ;
recv_2(D,C,receivedOtp1)          ;

match (receivedOtp1,otp1)         ;
  // Ok, device auth successful
  macro hiddenKeyBroadcast =
    HiddeBroadCastKey(signature,
                  broadCastKey)   ;
  // to hide the broadcast
  send_3(C,D,(otp2
            ,hiddenKeyBroadcast)) ;
  recv_4(D,C
  ,{securedPacket}keyAuthAndEnc)  ;

claim (C, SKR, keyAuthAndEnc)     ;
claim (C, SKR, broadCastKey)      ;
claim (C, Alive)                  ;
claim (C, Weakagree)              ;
claim (C, Niagree)                ;
claim (C, Secret, securedPacket)  ;
}} ;
```

The protocol label is "OCARIAuthAndEncProtocol","function" and "macro" are respectively used for the function definition and the formulas abbreviation. We have two roles played by the device (D) and the CPAN (C). The different transmissions are defined by the two events "send" and "receive". The "$\_id$" represents the event label, which is the identifier that links a "send" event with the appropriate "receive" event of the other role. The "match" event is used to model the equality tests.

The claim event types are the goals of the formal validation. For the secrecy of transmissions we use the claim "Secret". In order to be more precise, the secrecy of the transmission of session keys is formalized by the SKR (Session Key Reveal) claim. In addition, we used three authentication claim types, which are "Alive", "Weakagree", and "Niagree". [12] explains the three authentication claims. We assume that A is the initiator and B the responder.

- We consider that a protocol guarantees to A aliveness of B if, whenever A completes a run of the protocol, apparently with B, then the latter has previously been running the protocol.
- We consider that a protocol guarantees to A weak agreement with B if, whenever A completes a run of the protocol, apparently with B, then the latter has previously been running the protocol, apparently with A.
- And finally, we consider that a protocol guarantees to A non-injective agreement with B on a set of data items (variables) if, whenever A completes a run of the protocol, apparently with B, then the latter has previously been running the protocol, apparently with A, and B was acting as responder in its run, and the two agents agreed

on the data values corresponding to all the data items.

Figure 4 shows the results of the execution of the previous code. We used the maximum number of runs (100 runs).

In the result, the first, second, and third columns of the screen-shot in Figure 4 represent respectively: the protocol name, the concerned role (D and C), and a unique identifier of the claim. The fourth column represents the claim type with the parameters. The two last columns (status and comments) show the result of the verification process (Fail or Ok), and a short description. The "No attack within bounds" should be interpreted as: "*Scyther* did not find any attacks by reaching the bound" [2]. As we can see, the validation proves that our protocol is safe and secure.

B. Real tests

Figure 5 shows the topology used, in order to test performances of our security protocol. The implementation of our code was realized on the real OCARI source code, written in C language.

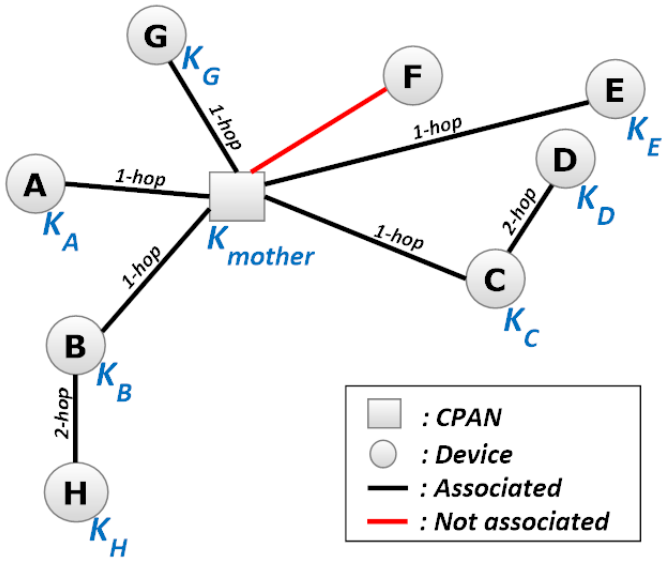

Figure 5: A real OCARI nodes topology

Each node represents a *Dresden Elektronik deRFsam3* 23M10-R3 device, having a 48 ko of RAM, 256 ko of ROM and a Cortex M-3 Processor. Each node X, except F, contains in prior a personalized key_{dX} generated from the key_i set up in the CPAN.

First, in order to know how much time our security mechanism takes during the association step, we tested the OCARI association time without security (we disable the security option), then with security. Table I presents the results of (1) a *single-hop association*, which means a direct association between the device and the CPAN. And of a *multi-hop association*, which means an association between a device and the CPAN using intermediate devices, called relays.

For capturing the exchanged messages we used a *Zolertia z1* sniffer (hardware) and *Wireshark*. The association time is equal to $timestamp\ of\ the\ authentication\ response\ minus\ (-)$ $timestamp\ of\ the\ association\ request$. The association of the device F is not accepted because F does not have the key_{dF}. One can note that the difference of the association time between the secured mode and the non secured one is small, and that the increase of the association time using security becomes less significant in the case of a multi-hop association. Thus, these results prove that our solution does not affect the network performances.

Mode	**single-hop**	**multi-hop**
Without security (average)	0,5243 (ms)	34,5076 (ms)
With security (average)	37,504 (ms)	45 (ms)

Table I: The association time of OCARI with and without security

Protocol	**processing time**
Our security protocol (average)	$\sim$ 3 (ms)
DTLS-Based protocol, 2048-bits keys (average) [10]	859 (ms) (<computation = 35 (ms)> + <Encrypt = 39 (ms)> + <signature = 726 (ms)> + <verification = 59 (ms)>)

Table II: Comparison between the security processing time of our protocol and the DTLS-based protocol

Then, in Table II, we compared our results with the DTLS-based protocol association time (described inII), realized by [10], and uses devices with similar capacity (Atmel SAM3U micro-controller and the Atmel AT97SC3203S TPM, with 48 kB of RAM). We did not take into account the number of the exchanged messages, that is equal to 4 messages in our security protocol and 6 messages at least in the DTLS-based protocol [10].

In order to compute the processing time of our solution ($challenge$, $otps$, and key_u generation + key exchange mechanism), we computed the association time of a 2-hop association without security which needs the exchange of 4 messages (*association request* $\times$ 2) + (*association response* $\times$ 2). Then we took the association time of 1-hop in the secured mode, that requires also the exchange of 4 messages (*association request* + *authentication request*) + *authentication response* + *association response*. By eliminating the time consumed by the messages exchange, a subtraction between the two association time represents the processing time of our security protocol.

It is true that the DTLS-based is very robust, however the difference between its processing time and our protocol processing time is very big. In addition, the fact that the DTLS-based protocol uses the asymmetric cryptography, consumes a lot of energy and requires an important memorization and processing capacity, which is not always adapted to limited devices.

Finally, Table III summarizes the differences between our solution (including the secure data exchange channel) and those discussed in Section II. We propose a score from 1

Claim				Status	Comments
OCARIAuthAndEncProtocol	D	OCARIAuthAndEncProtocol,D1	SKR keyGeneration(k(D,C),challenge)	Ok	No attacks within bounds.
		OCARIAuthAndEncProtocol,D2	SKR RevealBroadCastKey(signatureFunction(keyGenera...	Ok	No attacks within bounds.
		OCARIAuthAndEncProtocol,D3	Alive	Ok	No attacks within bounds.
		OCARIAuthAndEncProtocol,D4	Weakagree	Ok	No attacks within bounds.
		OCARIAuthAndEncProtocol,D5	Niagree	Ok	No attacks within bounds.
		OCARIAuthAndEncProtocol,D6	Secret securedPacket	Ok	No attacks within bounds.
	C	OCARIAuthAndEncProtocol,C1	SKR keyGeneration(k(D,C),challenge)	Ok	No attacks within bounds.
		OCARIAuthAndEncProtocol,C2	SKR RevealBroadCastKey(signatureFunction(keyGenera...	Ok	No attacks within bounds.
		OCARIAuthAndEncProtocol,C3	Alive	Ok	No attacks within bounds.
		OCARIAuthAndEncProtocol,C4	Weakagree	Ok	No attacks within bounds.
		OCARIAuthAndEncProtocol,C5	Niagree	Ok	No attacks within bounds.
		OCARIAuthAndEncProtocol,C6	Secret securedPacket	Ok	No attacks within bounds.

Figure 4: Formal validation results

Options	Rapidity	Authentication	PA1	Lightness	Energy efficiency	PA2	Confidentiality	Integrity	Final score
Our security protocol (using AES-GCM/CCM)	4	4	2	5	4	5	5	4	33
Our old solution [7] (using HMAC)	3	4	2	4	3	2	0	4	22
DTLS-Based protocol (2048-bits keys) [10]	1	5	5	0	1	5	5	5	27
HIP protocol [13]	4	3	3	3	3	1	5	4	26
Wi-Fi based IoT security protocol [16]	5	4	0	5	5	0	4	5	28

Table III: Comparison between the different security protocols

to 5 for each feature. The sum of the scores represents an evaluation score for the security approach. PA1: means Protection from the Denial of service attack, and PA2: means Protection from the cryptanalysis attack.

With these results, we can see that our security approach is the most adapted to the security of the WSNs and the IoT systems in general.

V. Conclusion and Future Works

In this work we designed a security protocol that enables to secure most of the WSNs thanks to its lightness and energy efficiency. It ensures a mutual authentication of the communicating entities and a protection of both the integrity and the confidentiality of the exchanged data. The "personalization" mechanism solves the problem of the internal identity usurpation. The proposed key management allows a safe and secure keys exchange between the concerned entities. Furthermore, this protocol provides a very fast establishment of a secure channel based on a robust, fast, and lightweight symmetric encryption algorithm (AES GCM/CCM). Finally, this solution is resilient against the cryptanalysis and the replay attacks. In our future works, we aim to create a secure communicating system between different CPANs and to facilitate a secure migration of devices from a network managed by a CPAN to a network managed by another CPAN.

References

[1] Marshall Abrams and Joe Weiss. Malicious control system cyber security attack case study–Maroochy Water Services, Australia. *McLean, VA: The MITRE Corporation*, 2008.

[2] Cas Cremers. Scyther. *Draft*, February 2014.

[3] Tim Dierks. The transport layer security (TLS) protocol version 1.2. 2008.

[4] Morris J Dworkin. SP 800-38C. Recommendation for block cipher modes of operation: The CCM mode for authentication and confidentiality. 2004.

[5] Morris J. Dworkin. SP 800-38D. Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC. Technical report, Gaithersburg, MD, United States, 2007.

[6] Mohamed T. Hammi, E. Livolant, P. Bellot, A. Serhrouchni, and P. Minet. MAC sub-layer node authentication in OCARI. In *2016 International Conference on Performance Evaluation and Modeling in Wired and Wireless Networks (PEMWN)*, pages 1–6, Nov 2016.

[7] Mohamed T. Hammi, E. Livolant, P. Bellot, A. Serhrouchni, and P. Minet. A lightweight mutual authentication protocol for the IoT. Technical report, 2017.

[8] Russell Housley, William Polk, Warwick Ford, and David Solo. Internet X. 509 public key infrastructure certificate and certificate revocation list (CRL) profile. Technical report, 2002.

[9] IEEE. IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs). IEEE Std 802.15.4-2011 (Revision of IEEE Std 802.15.4-2006), September 2011.

[10] Thomas Kothmayr, Corinna Schmitt, Wen Hu, Michael Brünig, and Georg Carle. {DTLS} based security and two-way authentication for the Internet of Things. *Ad Hoc Networks*, 11(8):2710–2723, 2013.

[11] Hugo Krawczyk, Ran Canetti, and Mihir Bellare. HMAC: Keyed-hashing for message authentication. 1997.

[12] Gavin Lowe. A hierarchy of authentication specifications. In *Computer security foundations workshop, 1997. Proceedings., 10th*, pages 31–43. IEEE, 1997.

[13] Francisco Vidal Meca, Jan Henrik Ziegeldorf, Pedro Moreno Sanchez, Oscar Garcia Morchon, Sandeep S Kumar, and Sye Loong Keoh. HIP security architecture for the IP-based internet of things. In *Advanced Information Networking and Applications Workshops (WAINA), 2013 27th International Conference on*, pages 1331–1336. IEEE, 2013.

[14] D M'Raihi, M Bellare, F Hoornaert, D Naccache, and O Ranen. HOTP: An HMAC-based one-time password algorithm. IETF, RFC 4226, December 2005.

[15] R. L. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-key Cryptosystems. *Commun. ACM*, 21(2):120–126, February 1978.

[16] Freddy K Santoso and Nicholas CH Vun. Securing IoT for smart home system. In *Consumer Electronics (ISCE), 2015 IEEE International Symposium on*, pages 1–2. IEEE, 2015.