Secure Communication for Smart IoT Objects: Protocol Stacks, Use Cases and Practical Examples

Riccardo Bonetto[¶], Nicola Bui[*†], Vishwas Lakkundi[†], Alexis Olivereau[‡], Alexandru Serbanati[§], Michele Rossi[*¶]

***Abstract*—In this paper we discuss security procedures for constrained IoT devices. We start with the description of a general security architecture along with its basic procedures, then discuss how its elements interact with the constrained communication stack and explore pros and cons of popular security approaches at various layers of the ISO/OSI model. We also discuss a practical example for the establishment of end-to-end secure channels between constrained and unconstrained devices. The proposed method is lightweight and allows the protection of IoT devices through strong encryption and authentication means, so that constrained devices can benefit from the same security functionalities that are typical of unconstrained domains, without however having to execute computationally intensive operations. To make this possible, we advocate using trusted unconstrained nodes for the offloading of computationally intensive tasks. Moreover, our design does not require any modifications to the protocol stacks of unconstrained nodes.**

***Keywords*-Information Security; Internet of Things; Smart Objects; 6LoWPAN; IPsec; IKEv2;**

I. Introduction

The Internet of Things (IoT) [1] has immense potential to change many of our daily activities, routines and behaviors. The pervasive nature of the information sources means that a great amount of data pertaining to possibly every aspect of human activity, both public and private, will be produced, transmitted, collected, stored and processed. Consequently, integrity and confidentiality of transmitted data as well as the authentication of (and trust in) the services offering that data is crucial. Hence, security is a critical functionality for the IoT.

Data networks, especially wireless, are prone to a large number of attacks such as eavesdropping, spoofing, denial of service and so on. Legacy Internet systems mitigate these attacks by relying on link layer, network layer, transport layer or application layer encryption and authentication of the underlying data. Though some of these solutions are applicable to the IoT domain, the inherently limited processing and communication capabilities of IoT devices prevent the use of full-fledged security suites.

[*]Consorzio Ferrara Ricerche, Ferrara, Italy. [†]Patavina Technologies, Padova, Italy. [‡]Commissariat à l'Energie Atomique et aux Energies Alternatives, Saclay, France. [§]CATTID, University of Rome "La Sapienza", Italy. [¶]DEI, University of Padova, Italy.

978-1-4673-1239-4/12/$31.00 ©2012 IEEE

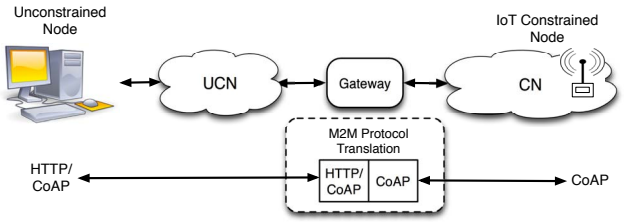

Figure 1. Communication scenario.

The setup of secure end-to-end channels is possible nowadays within the unconstrained network (UCN) domain through a number of mature technologies such as IPsec [2], SSL/TLS [3] or DTLS [4], which however cannot be directly leveraged by constrained network (CN) nodes due to memory space and processing power constraints. To solve these problems, we discuss in this paper a suitable security architecture for IoT with the following objectives: 1) security suites currently employed within UCNs shall continue to be used with no modifications on the UCN side, 2) originating security handshakes/procedures are handled differently within the CN so that constrained nodes can handle their complexity, thus being able to establish end-to-end secure channels, whereas 3) unconstrained nodes shall not notice any deviation from their standard procedures.

The solution discussed here is based on the offloading of computationally intensive tasks to a trusted and unconstrained node; this node is then responsible for the calculation of the master session key on behalf of the constrained IoT devices under its purview. IoT Gateways (GW), placed on the edge between UCN and CN as shown in Fig. 1, have the role of adapting the communication between these two domains and could as well be used for this purpose. In fact, their role usually involves the adaptation between different protocol-layer implementations, which entails physical (PHY) and link (LL) layers but could also encompass all layers up to and including the application layer. The fact that GWs are generally UCN devices means that they can also be used for *scaling down the functionalities* (including security) from the UCN to the CN domain and also for managing security settings in peripheral (i.e. CN) networks. In order to maintain the end-to-end approach, GWs have to be invisible from the viewpoint of the communicating endpoints.

Despite end-to-end security, lower layers may continue using heterogeneous security features across network sub-domains or for point-to-point communication.

This paper is organized as follows: Section II defines the building blocks of the security architecture and the pertinent security procedures, Section III elaborates use cases, security scenarios and describes a practical security example and finally, our conclusions are drawn in Section IV.

II. Security Architecture

A. Building Blocks

The architecture of the proposed IoT communication protocol stack is illustrated in Fig. 2, with the functional security blocks shown on the left. Evidently, security functionalities and the communication stack (right) are closely knit with each other.

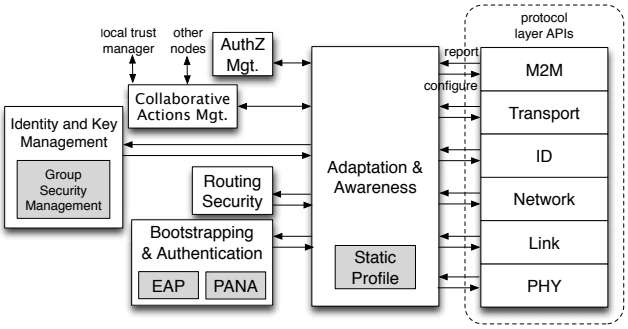

Figure 2. Secure Protocol Architecture for IoT.

Machine-to-Machine (M2M) layer resides just below the application layer of IoT devices. It addresses the lack of interoperability of current M2M technology and makes communication between different network elements possible either through the adoption of a common descriptive language or through message translation. It is also referred to as service layer. Our design includes a suitable translation proxy, which may reside in the gateway [5] as shown in Fig. 1.

Transport (TRA) layer provides end-to-end performance guarantees between communicating endpoints, especially in terms of orderly delivery and reliability. A TCP-like transport is optional for IoT devices and depends on the required end-to-end Quality of Service (QoS); refer to Section III-A for further details.

Identification (ID) layer is an addition to standard unconstrained protocol stacks. It is designed to carry out the resource identification task within the IoT domain, that is hitherto been performed by IP addresses in their dual role of being both identifiers and network locators. Advantages of this approach include enhanced security, which can be embedded in routing and be coupled with an authentication service based on the node ID. The main drawbacks include the lack of a widely adopted protocol suite in the Internet and an increased protocol stack footprint. Considering its pros and cons, we keep this option open in our design by making it *non-mandatory*. The Host Identity Protocol (HIP) is a notable technological example for this concept [6].

Network (NET) layer houses the Internet protocol, takes care of node addressing and packet routing. We foresee IPv6 [7] as the most suitable Internet technology. As we shall see shortly, some adaptation is required for the constrained domain, as compression of long IPv6 headers is beneficial to the transmission of IPv6 datagrams over CNs.

Medium Access Control (MAC) layer is responsible for channel access, thereby determining how devices schedule their transmissions over the physical layer medium.

Physical Layer (PHY) deals with modulation/demodulation, channel coding and transmission over the given medium. PHY is technology specific. Our aim is to make systems interoperable by filling the gaps that prevent different M2M technologies from communicating with each other. Therefore, PHY and MAC layer design issues are outside the scope of our architectural design.

The security blocks and their functions are as follows:

Bootstrapping and Authentication controls the network entry of nodes. Authentication is highly relevant to IoT and is likely to be the first operation carried out by a node when it joins a new network, for instance, after mobility. It is performed with a (generally remote) authentication server using a network access protocol such as the Protocol for Carrying Authentication for Network Access (PANA) [8]. For greater interoperability, the use of the Extensible Authentication Protocol (EAP) [9] is envisioned. Upon successful authentication, higher layer security associations could also be established (such as IKE followed by IPsec [2]) and launched between the newly authenticated endpoint and the access control agent in the associated network.

Static Profile represents the knowledge by an endpoint of its own resources (such as identity, battery, computing power, memory size, etc.) and the security settings it intends to use or needs from the network. The static profile can be read-only (preset by vendor), write-once (set by manufacturer) or rewritable (user enabled). Note that certain security primitives may be computationally prohibitive for IoT objects; a negotiation is thus required before the establishment of a secure channel so that the concerned endpoints can agree upon a cryptographic suite.

Collaborative Actions Management is invoked whenever a node cannot fulfill by itself a task it has to accomplish, for instance, when the task is too computationally intensive for the resource constrained IoT node. It interacts with a *trusted entity* in the CN topology to learn about possible assisting peers. This building block is particularly relevant to the proposal in this paper, as we detail in Section III-C. Also, refer to [10] for recent cooperative key establishment strategies.

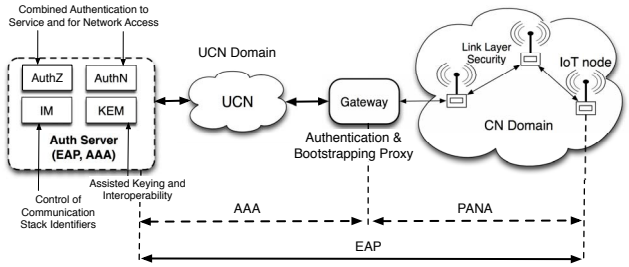

Figure 3. High-level view of the Network Entry procedure.

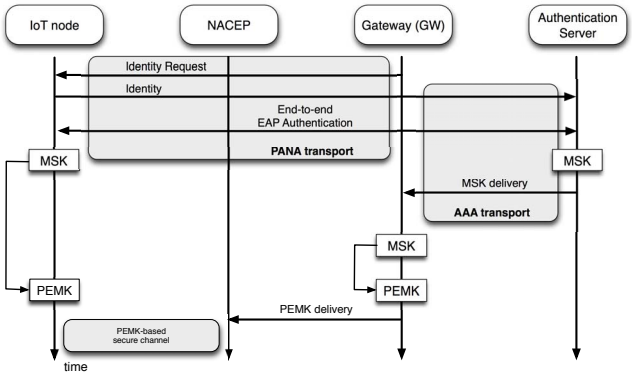

Figure 4. Flow Diagram of the Network Entry procedure.

Identity and Key Management block ensures secure interaction between endpoints. It establishes node privacy by choosing a particular identity (or pseudonym) for use in the communication stack. It provides secure communications by agreeing on a key with a peer, for instance, via a dedicated Authenticated Key Exchange (AKE) protocol, which may be bound to the identities in use (as in the HIP case).

Adaptation and Awareness block is responsible for configuring the protocol stack of the IoT node and gathering information on its current status. While interacting with the IP layer, for example, it would push security parameters (keys) to IPsec. It would also push IP addresses (or at least, suffixes) through the Identity and Key Management block. The Awareness part of the module contains knowledge about the status of current node and its capabilities (i.e., its Static Profile).

Group Security Management is responsible for enforcing multicast security at the IP layer.

Routing Security block implements a protocol solution aimed at mitigating classical routing attacks. This module is likely to have relationships with Bootstrapping and Authentication and Local Trust Manager. These interactions are assumed to take place through the Adaptation and Awareness module. Routing Security can be instantiated through the implementation-dependent parameters of the routing protocol in use, e.g. RPL [11].

Authorization Management (AuthZ Mgt.) manages inbound and outbound access to Services, interacting with the existing Authorization infrastructure in order to retrieve certificates for accessing other resources and to verify whether authenticated users are authorized to access own resources when they do not use certificates.

B. Security Procedures

Further, we illustrate two relevant security procedures, namely, Network Entry and Secured Connection to a Distant Peer. The first procedure is used for authentication of an IoT node to either a remote server or a host, whereas in the second procedure an IoT node establishes a secure channel with an endpoint residing outside the CN domain.

Network Entry: It defines how a constrained IoT node authenticates to a remote server through a neighbor and the Gateway connecting the CN domain to the UCN domain as shown in Fig. 3. An EAP-based authentication [9] is exploited between the IoT node (on the left) and a remote Authentication Server (supporting EAP, referred to as "Auth Server" in the figure). EAP is carried by the Protocol for carrying Authentication for Network Access (PANA) between the IoT node and the Gateway, and by a AAA protocol between the Gateway and the Authentication Server. As a result of successful authentication, a Master Session Key (MSK) between the IoT node and the Authentication Server is established and then carried from the Authentication Server to the Gateway over AAA. A PEMK secret is then derived from this MSK at the Gateway and delivered to the Network Access Control Enforcement Point (NACEP). PEMK bootstraps the security association between the IoT Node and the NACEP. The flow diagram of the Network Entry procedure is shown in Fig. 4.

Secured Connection to a Distant Peer: For the following discussion we refer to Fig. 1. An initial possibility is to *split* the end-to-end channel into two segments: S1) from the IoT node to the Gateway and S2) from the Gateway to the unconstrained node. Thus, two secure channels are set up, for segments S1 and S2, respectively. For S1, different alternatives are possible, ranging from dedicated and customized protocols to hop-by-hop security, which is obtained as a sequence of secure links, through the use of appropriate link layer security mechanisms. Note that hop-by-hop security might be the only possible option for S1 when IoT nodes are unable to authenticate with each other. Hence, from the Gateway to the unconstrained node any existing security association can be used, i.e. IPsec, SSL/TLS, DTLS, etc. The Gateway will thus perform the needed protocol translation, by decoding the data coming from S1 and re-encoding it into S2, according to the security policy/credentials used within each segment. This allows one to tailor the security level to each segment. A second option consists in having the initiating IoT node agree upon a shared secret key mechanism with the remote (unconstrained) peer. This procedure requires the preliminary establishment of a

secure channel from the IoT node and the Gateway and is further exemplified in Section III-C, where we discuss a practical procedure to set up secure end-to-end channels using IPsec. According to this second case, once the shared secret is established, the IoT node communicates directly to the remote peer using a single end-to-end secure channel (one security policy).

III. Application Examples

In this section, we illustrate the use cases and practical examples pertaining to secure communications taking into account the protocol stack architecture illustrated earlier in Section II.

A. Protocols Stacks for IoT Communication

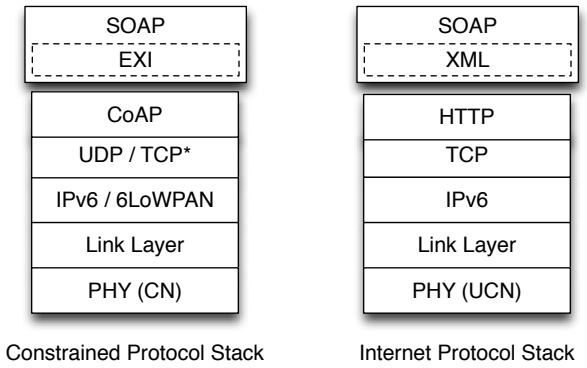

Figure 5. IETF CoAP-based protocol stack.

Fig. 5 shows a possible protocol stack for IoT devices based on the Constrained Application Protocol (CoAP) [12] (left) along with the corresponding protocol stack for unconstrained devices (right). IPv6 over low power wireless personal area networks (6LoWPAN) [13] defines how IPv6 addressing can be efficiently implemented within the CN. The transport layer is expected to exploit UDP, however an optimized transport solution can be optionally used if the application so requires. In Fig. 5, TCP* refers to a dedicated transport protocol, different from TCP and optimized for use within CNs. In fact, standard TCP is highly inefficient for CNs due to their typically large bandwidth-delay product, high packet losses and their peculiar traffic model, that will be likely composed of tiny packets originating from request/response pairs. Simple Object Access Protocol (SOAP) [14] can be used for exchanging structured information at the application layer, relying on Efficient XML Interchange (EXI) on the CN side [15].

CoAP is an application-layer protocol that is being defined by the IETF CoRE working group to enable M2M interactions among constrained communicating objects. It is lightweight and easily mappable onto HTTP [5]. This promising approach effectively considers the pervasiveness of HTTP in the Web. Enabling HTTP communication over CN environments will further extend CoAP's applicability. The publish-subscribe paradigm has been adopted as the design choice for CoAP in the hope that its success for the Web 2.0 technology will ease its wider adoption on the market. According to this paradigm, end devices behave as Web servers and exploit a limited number of messages, namely, GET, POST, DELETE and UPDATE. CoAP is intended to replace HTTP in CN environments. HTTP can be used in the UCN part of the network, whereas CoAP is indicated for use within the CN domain, which in turn requires the presence of proper HTTP/CoAP translation proxies [5] somewhere in the UCN (note that the proxy functionality can also be placed in the gateway between the UCN and the CN).

B. Security Considerations

Regarding communication security, CoAP does not itself provide protocol primitives for authentication or data encryption. Wherever required, they can be provided by a secure communication protocol such as IPsec [2] (network layer) or DTLS [4] (transport layer) or object security (within the payload). IPsec works at the network layer and permits the establishment of end-to-end authenticated and encrypted channels between endpoints. Datagram Transport Layer Security (DTLS) operates at the transport layer providing analogous security features. It builds on Secure Sockets Layer (SSL) technology but assumes UDP as the transport protocol. Endpoints that require authorization for certain operations are expected to include one of these two types of security.

IPsec ***vs*** **DTLS:** Each technology has its own pros and cons. Technically, both IPsec and SSL are mature technologies providing all we need in terms of security features, although they reside in different layers of the ISO/OSI stack. Pros and cons of these technologies lie rather in the limits of existing implementations and in their usage models. For IPsec, we note that existing implementations are hardly compatible with each other and often require some manual configuration; in fact, most IPsec solutions for setting up Virtual Private Networks (VPN) require third-party hardware and/or software. Moreover, in order to access an IPsec VPN, a given endpoint must have an IPsec client application installed. This is both an asset and a drawback. The former resides in the fact that a client machine, in order to gain access to a given VPN, is required not only to have a compatible IPsec client installed, but also to have it properly configured (and this entails a further level of security). The drawbacks are that the maintenance of the client software (with valid licenses) on all required client machines can be expensive and the installation can be quite complex and may even require some human intervention. Conversely, SSL technology is nowadays implemented in nearly all Web browsers and has reached a good maturity level; its open source implementations also exist (e.g. OpenSSL, http://www.openssl.org/) and they interoperate satisfactorily. This

means that almost every computer in the world is already equipped with the necessary software to connect to an SSL VPN. Another upside of SSL VPNs is that they allow more precise access control. In fact, operation at the transport layer (SSL/TLS or DTLS) allows VPN tunnels to be allocated to specific applications rather than to a specific machine (or often to an entire LAN). So, users on SSL VPN connections can only access the applications that they are configured to access rather than an entire network. Probably, the major drawback of SSL technology is that they only work for Web-based applications and as such they do not always provide native access to all network resources and SSL VPNs are not usable for file sharing or backup operations. This use is not impossible but requires the addition of SSL support onto the application(s) of interest, which somewhat increases its(their) complexity. It is still unclear as to which technology is the best for IoT, although recent developments from IETF seem to encourage the adoption of DTLS.

In the next subsection, we propose a lightweight methodology to establish an end-to-end secure channel, requiring minimal involvement of constrained IoT devices, while keeping all protocol operations unchanged within the UCN. This is achieved through the delegation of some of the steps associated with channel establishment to a trusted Gateway (GW). Our proposal considers an IPsec-based security association and can be seen as a particular instance of the security architecture of Section II. Note, however, that a totally similar procedure would also work for DTLS.

C. An IPsec-based Security Association Example

Here, we discuss an example for the establishment of an end-to-end secure communication channel between a Constrained Device (CD) and an Unconstrained Device (UD) with the following assumptions:

1) The constrained node uses 6LoWPAN for addressing and CoAP as the application layer protocol.
2) The unconstrained node uses IPv6 for addressing and HTTP as the application layer protocol.
3) The constrained node is already authenticated with the Gateway (GW).
4) There exists a security policy allowing secure communications within the constrained network domain (and in particular between GW and CD).
5) The gateway is a trusted entity.

Our proposal makes it possible for an unconstrained node to set up an IPsec-ESP Transport Mode [16] connection with an IoT device while moving the master session key generation and authentication processes from the IoT node to the trusted gateway. The ESP mode, which provides data encryption and authentication, allows the setup of an end-to-end secure connection between two peers by encrypting the payload, while leaving the IPv6 headers untouched. The cryptographic keys are generated and exchanged according to the IKE protocol [17] using the Elliptic Curve Diffie Hellman key exchange scheme [18], [19]. With the aid of these mechanisms, the logic for key generation and authentication is moved from the IoT node to the corresponding GW, thus relieving the IoT device from the computational burden associated with the generation of cryptographic data. Fig. 6 illustrates the relevant steps involved in the procedure.

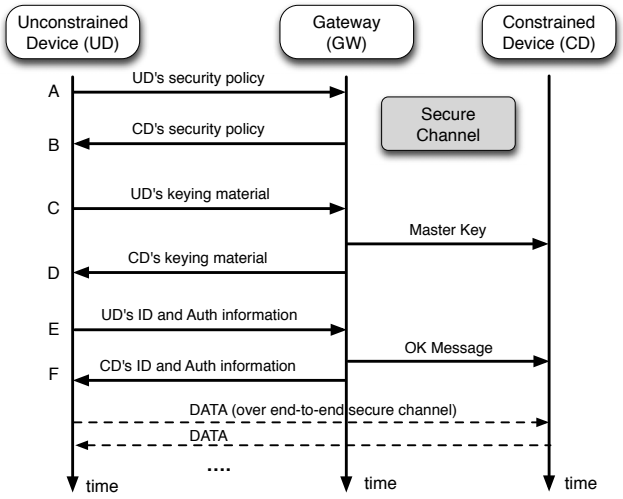

Figure 6. Lightweight IPsec security association.

The UD sends a service request to the CD (message A in Fig. 6), including the desired security level. This request is routed through the Internet and the GW. The GW intercepts the request and acts as if it were the desired Constrained Device (CD, see right hand side in Fig. 6). Specifically, it responds on behalf of CD (taking into account its Static Profile), by sending a message B which includes the security policy supported by the constrained device (which should be a valid subset of that indicated by UD through message A). Upon receiving message B, UD generates the initial keying material for the the establishment of an IPsec-ESP Security Association and sends it along with a further message C. Thus, once this message C is received at the GW, the latter generates the keying material (for UD) on behalf of the CD, including it into a further message D that is sent from the GW to the unconstrained device UD. Thus, GW sends the master key to the constrained device exploiting a secure channel (that needs to be established beforehand). Upon receiving message D, UD obtains its own master key, while the constrained terminal CD has its master key dispatched in a secure way by the gateway after receiving message C. Once the key exchange phase is over, the authentication credentials are exchanged between the unconstrained node and the gateway. Messages E and F are used to authenticate the two peers (this is performed over an encrypted channel, using the shared secret key that both endpoints obtain from messages A,B,C and D). If GW accepts UD's authentication credentials, an OK message is sent to the CD. Upon the

receipt of this OK message, UD and CD can communicate with each other by exploiting the channel secured by IPsec-ESP.

During the communication phase (DATA), in case UD and CD respectively use HTTP and CoAP, the gateway needs to perform CoAP/HTTP translations. Due to this, the gateway must be able to decrypt the packets exchanged by the communicating endpoints. Hence, the assumption that the gateway is trusted and that it knows the master keys for the active connections does not appear to be excessively bold in this case. When instead CoAP is used by both endpoints the Gateway does not necessarily have to decrypt the exchanged DATA packets; from this point on, GW becomes fully transparent and the secure flow can be redirected to any other Gateway without affecting security and connectivity. Note that the translation from IPv6 to 6LoWPAN does not require the gateway to decrypt the messages flowing through it, as shown in [20]. This latter case is particularly interesting as it allows the exploitation of a trusted Gateway in the initial setup phase, by effectively establishing an end-to-end channel, for which no state is maintained at the edge IoT Gateway.

Note that the above procedure entails the offloading of some of the computational burden, namely the generation of the master key, from the IoT device on to a suitable (trusted) Gateway node. The latter is thus required to act on behalf of the constrained devices under its jurisdiction, impersonating them during the initial IPsec handshakes. A totally similar procedure is executed in the opposite direction, i.e., when CD is the initiator.

D. Formal Protocol Description

The IKEv2 [17] secure connection setup is made of two phases:

1) `IKE_SA_INIT`;
2) `IKE_AUTH`.

The first phase is responsible for negotiation of the cryptographic primitive, exchange of nonces and the Diffie Hellman exchange procedure. The second phase is responsible for authentication of the messages exchanged during the first phase and the exchange of the identifiers (and optionally of the certificates).

Terminology: let i be the Initiator (the unconstrained terminal) of the IKE packets flow and let r be the Responder (the gateway). Let `HDR` be the header of each IKE packet. The keys generated using the Diffie Hellman key are called `SK`, and initially there are seven of such keys:

- `SK_d`: key used to generate further keying material when needed;
- `SK_ai`: Initiator's authentication key;
- `SK_ar`: Responder's authentication key;
- `SK_ei`: Initiator's encryption/decryption key;
- `SK_er`: Responder's encryption/decryption key;
- `SK_pi`: Initiator's key to encrypt the `IKE_AUTH` payload;
- `SK_pr`: Responder's key to encrypt the `IKE_AUTH` payload.

Let `[...]` denote an optional field in a packet. Let `prf` denote a pseudo-random function on which the two peers agree using the `SA` field. Let `SPI` denote the Security Parameter Index which allows a peer to identify one particular Security Association (`SA`) in its database. Moreover, let `N` be a nonce, `SKEYSEED=prf(`N_i, N_r, g^{ir}`)` and `{SK_d|SK_ai|SK_ar|SK_ei|SK_er|SK_pi|SK_pr}=prf+(SKEYSEED, Ni|Nr|SPIi|SPIr)`. In the last equation, `prf+(.)` denotes the iteration of the pseudorandom function according to the following rule: `prf+`$(K, S) = T_1|T_2|T_3|\ldots$ where T_1 =`prf`$(K, S|0x01)$ and T_i =`prf`$(K, T_{i-1}|S|0x0i)$ $\forall i > 1$. The `SKEYSEED` is computed by GW and sent to CD, after which CD computes all the keying material as described above.

Protocol description: Each phase of the protocol requires two packets: a proposal from the initiator and a response from the responder. The packets exchanged during the two phases are (with reference to Fig. 6):

message C ($i \rightarrow r$)**:**
`HDR,SAi1,KEi,Ni;`
message D ($r \rightarrow i$)**:**
`HDR,SAr1,KEr,Nr;`
message E ($i \rightarrow r$)**:**
`HDR,SK{IDi,[CERT,][CERTREQ,][IDr,]AUTH,SAi2,TSi,TSr};`
message F ($r \rightarrow i$)**:**
`HDR,SK{[CERT,]AUTH,SAr2,TSi,TSr}.`

It is worth noting that the messages A and B shown in Fig. 6 are included in messages C and D when using IKEv2. The first two packets C and D make it possible to agree on a shared secret (the Diffie Hellman key), which is used to generate all the subsequent keying material. The last two packets E and F allow the two peers to authenticate with one another and then to start a secure end-to-end communication.

IKE Authentication: Each peer signs (or authenticates using a Message Authentication Code (MAC)) its packet during the `IKE_AUTH` phase:

- the initiator authenticates its packet using `SK_pi` starting from the first byte of the `SPIi` in the header to the last byte of the packet concatenated to `Ni|prf(SK_pr,IDr')`;
- the responder authenticates its packet using `SK_pr` starting from the first byte of the `SPIi` in the header to the last byte of the packet concatenated to `Nr|prf(SK_pi,IDi')`.

The signature (or the MAC) is computed using the algorithms specified in the AUTH field in the authentication payload of the first two packets. The initiator and the responder are not required to use the same cryptographic algorithm for authentication (for example, the initiator can sign its packet while the responder can use a MAC).

The security of the protocol lies on the strength of the cryptographic algorithms chosen by the peers. For a security analysis of the IKE protocol, the reader is referred to [21].

IV. Conclusions

In this paper, we have presented a security architecture for IoT, discussing pros and cons of popular security approaches for the implementation of security suites and the establishment of secure communication channels. Furthermore, we have discussed means to embed security functionalities into IoT protocol stacks and we have proposed a lightweight procedure to set up secure end-to-end channels between unconstrained (and remote) peers and IoT devices. While the protocols used are widely employed in unconstrained networks, we ascertain that this procedure is lightweight because the gateway supports part of the computational load for performing cryptographic tasks instead of the constrained device. Our current work is devoted to the refinement of the latter procedure, along with its implementation and experimentation in real testbeds.

Acknowledgment

This work has been supported in part by the European Commission through the FP7 EU project titled "Internet of Things–Architecture (IoT-A)" (G.A.no.257521, http://www.iot-a.eu/public).

References

[1] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A Survey," *Elsevier Computer Networks*, vol. 54, no. 15, Oct. 2010.

[2] S. Frankel and S. Krishnan, "RFC 6071: IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap," IETF Request For Comments, http://tools.ietf.org/rfc/rfc6071.txt, Feb. 2011.

[3] T. Dierks and E. Rescorla, "RFC5246: The Transport Layer Security (TLS) Protocol Version 1.2," IETF Request For Comments, http://tools.ietf.org/rfc/rfc5246.txt, Aug. 2008.

[4] E. Rescorla and N. Modadugu, "RFC6347: Datagram Transport Layer Security Version 1.2," IETF Request For Comments, http://tools.ietf.org/rfc/rfc6347.txt, Jan. 2012.

[5] A. Castellani, S. Loreto, A. Rahman, T. Fossati, and E. Dijk, "IETF Draft: Best Practices for HTTP-CoAP Mapping Implementation," IETF draft-castellani-core-http-mapping-03, Mar. 2012.

[6] R. Moskowitz and P. Nikander, "RFC 4423: Host Identity Protocol (HIP) Architecture," IETF Request For Comments, http://tools.ietf.org/rfc/rfc4423.txt, May 2006.

[7] S. Deering and R. Hinden, "RFC2460: Internet Protocol, Version 6 (IPv6) Specification," IETF Request For Comments, http://tools.ietf.org/rfc/rfc2460.txt, Dec. 1998.

[8] D. Forsberg, Y. Ohba, B. Patil, H. Tschofenig, and A. Yegin, "RFC5191: Protocol for Carrying Authentication for Network Access (PANA)," IETF Request For Comments, http://tools.ietf.org/rfc/rfc5191.txt, May 2008.

[9] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, and H. Levkowetz, "RFC 3748: Extensible Authentication Protocol (EAP)," IETF Request For Comments, http://www.ietf.org/rfc/rfc3748.txt, Jun. 2004.

[10] Y. B. Saied, A. Olivereau, and D. Zeghlache, "Energy Efficiency in M2M Networks: A Cooperative Key Establishment System," in *IEEE International Congress on Ultramodern Telecommunications and Control Systems (ICUMT)*, Budapest, Hungary, Oct. 2011.

[11] T. Winter *et.al.*, "RPL: IPv6 Routing Protocol for Low power and Lossy Network," IETF (Work in Progress) draft-ietf-roll-rps-19, Mar. 2011.

[12] Z. Shelby, K. Hartke, C. Bormann, and B. Frank, "Constrained Application Protocol (CoAP)," IETF (Work in Progress) draft-ietf-core-coap-09, Mar. 2012.

[13] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, "RFC 4944: Transmission of IPv6 Packets over IEEE 802.15.4 Networks," IETF Request For Comments, http://www.rfc-editor.org/rfc/rfc4944.txt, Sep. 2007.

[14] M. Gudgin, M. Hadley, N. M. J.-J. Moreau, H. F. Nielsen, A. Karmarkar, and Y. Lafon, "SOAP Version 1.2 Part 1: Messaging Framework," Apr. 2007, W3C Recommendation.

[15] A. P. Castellani, M. Gheda, N. Bui, M. Rossi, and M. Zorzi, "Web Services for the Internet of Things through CoAP and EXI," in *IEEE RWFI*, Kyoto, Japan, Jun. 2011.

[16] S. Kent, "RFC4303: IP Encapsulating Security Payload (ESP)," IETF Request For Comments, http://tools.ietf.org/rfc/rfc4303.txt, Dec. 2005.

[17] C. Kaufman, P. Hoffman, Y. Nir, and P. Eronen, "RFC 5996: Internet Key Exchange Protocol Version 2 (IKEv2)," IETF Request For Comments, http://www.ietf.org/rfc/rfc5996.txt, Sep. 2010.

[18] E. Rescorla, "RFC 2631: Diffie-Hellman Key Agreement Method," IETF Request For Comments, http://tools.ietf.org/rfc/rfc2631.txt, Jun. 1999.

[19] D. J. E. Barker and M. Smid, "Recommendations for pairwise key establishment schemes using discrete logarithm cryptography (revised)," FIPS SPECIAL PUB 800-56A, Mar. 2007.

[20] S. Raza, S. Duquennoy, T. Chung, D. Yazar, T. Voigt, and U. Roedig, "Securing Communication in 6LoWPAN with Compressed IPsec," in *IEEE International Conference on Distributed Computing in Sensor Systems (IEEE DCOSS 2011)*, Barcelona, Spain, Jun. 2011.

[21] C. Cremers, "Key exchange in IPsec revisited: formal analysis of IKEv1 and IKEv2," in *European conference on research in computer security (ESORICS)*, Leuven, Belgium, Sep. 2011.