Computer Networks 64 (2014) 273–295

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Lightweight collaborative key establishment scheme for the Internet of Things

Yosra Ben Saied [a,*], Alexis Olivereau [a], Djamal Zeghlache [b], Maryline Laurent [b]

[a] CEA, LIST, Communicating Systems Laboratory, F-91191 Gif-sur-Yvette, France
[b] Telecom SudParis, 91011 Evry, France

ARTICLE INFO

Article history:
Received 14 May 2013
Received in revised form 27 December 2013
Accepted 2 February 2014
Available online 18 February 2014

Keywords:
Internet of Things
Wireless sensor networks
Key establishment
End-to-end security
Energy efficiency

ABSTRACT

This work addresses new security issues in the Internet of Things (IoT). The heterogeneous nature of IoT communications and imbalance in resource capabilities between IoT entities make it challenging to provide the required end-to-end secured connections. Clarifying how existing security protocols can be adapted to fulfill these new challenges still has to be improved. A direct use of existing key exchange schemes between two IoT entities may be unfeasible unless both entities be able to run the resource consuming cryptographic primitives required to bootstrap them – thus leaving aside a whole class of resource-constrained devices. In this paper, we revisit existing end-to-end security standards and key establishment schemes and discuss their limitations considering the specific scenarios of the IoT. Later, we propose novel collaborative approaches for key establishment designed to reduce the requirements of these existing security protocols. A constrained device may delegate its heavy cryptographic load to less constrained nodes in neighborhood exploiting the spatial heterogeneity of IoT environment. We demonstrate through a performance analysis that our collaborative key establishment solution allows for a reduction in energy consumption at the constrained device by up to 80% in comparison with existing key establishment schemes.

1. Introduction

The current transition from legacy Internet to Internet of Things (IoT) involves multiple changes in its communication paradigms. The diversity of scenarios where internetworked entities have to exchange information with each other without human interaction is increasing and is planned to extend to almost all environments, from individual customers' everyday life to industrial processes. Accordingly, more and more objects become able to communicate, following as a rule of thumb an always greater interaction with the physical world, which is not only timely and accurately sensed but also understood and acted upon. Wireless sensor networks [1] were the first step in this direction. Machine to machine (M2M) communications [2] largely extending the sensor networking model, represents a more advanced type of network referring to data communication between physical devices without human intervention. M2M systems adopt a distributed communication model wherein any two nodes may establish relationship with each other, provided that one is offering the service, or resource, which is needed at the other end. To that respect, M2M systems broke the logical and topological simplicity of sensor networks. Contrary to what happens in WSNs, the communication path between two nodes does not have to follow a hierarchical path, e.g., from sensor to sink, and from sink to remote management units. A sensor node in an M2M environment

* Corresponding author. Tel.: +33 678508204.
E-mail addresses: yosra_bensaied@yahoo.fr, yosra.ben-saied@cea.fr (Y.B. Saied).

http://dx.doi.org/10.1016/j.comnet.2014.02.001
1389-1286/

will likely have direct communications with other peers irrespective of their distance, role and capabilities, provided that these relationships are desirable from the viewpoint of the M2M scenario. This novel paradigm, wherein nodes communicate with a large set of heterogeneous entities through a decentralized pattern, leads to situations where unbalanced resource capabilities between the two communicating peers are confronted.

The IoT further extends the M2M paradigm into two directions. First, it aims to interconnect much wider sets of objects, even those that were not natively supposed to be able to communicate. Barcodes and tags allow otherwise inert objects to advertise their presence and sometimes to receive and store information. This makes them part of the connected world. Second, the IoT targets universality and global interoperability whereas most M2M architectures are dedicated to the fulfillment of a given task, be it wide-scale (e.g., Smart Grid operation [3]) or small-scale (e.g., home automation [4]). The advantages of interconnecting huge sets of "things" belong to the fields of adaptation (ability to sense/act on the environment) and autonomous orchestration of new services (interactions appear when entities discover each other, along with their needs and capabilities).

In order to securely accomplish this integration, end-to-end communications between heterogeneous nodes have to be established as required by the decentralized characteristic of IoT scenarios. However, the prerequisite for any secure channel setup, that is, key establishment, could be either unaffordable or prohibitively expensive for a wide range of nodes, found in the IoT environment, which precisely exhibit constraints in both computing power and battery capacity. While latency could be induced, this is not where the main problem lies: a key establishment operation occurs indeed at the beginning of a novel communication without affecting it afterwards, except when rekeying is needed. For example, a lengthy key establishment phase, in the order of a few seconds or dozens of seconds, would still be acceptable if it occurred only once a day. More critical are the consequences in terms of energy consumption. Battery-powered sensor nodes can be disseminated in hazardous environments. Some are built-in within products and are expected to have at least the same lifetime as their hosts. Changing a discharged battery could therefore be either demanding, or unacceptable.

To the best of our knowledge, the design of efficient key establishment protocols that clearly address heterogeneous IoT communications between peers with different resource capabilities is not undertaken yet. In the literature, energy efficiency was an important concern in WSNs because of the low capabilities of sensor nodes. Unfortunately IoT requirements go far beyond those of WSNs, since it is assumed in these latter that the sensor nodes are isolated from the Internet and connected to external hosts via dedicated gateways. Efficient key establishment schemes proposed in traditional WSNs are not targeting a secure end-to-end communication between the sensor node and remote hosts. Instead, they discuss security of communications within the sensor network. In this paper, we take into account the inadequacies of these proposals as well as the IoT requirements to design new key establishment schemes able to enable end-to-end secure communications between nodes with different resource capabilities, in the context of the IoT.

We propose to exploit the heterogeneity of IoT nodes to involve unconstrained ones in a collaborative key establishment process, wherein they would make available to otherwise constrained peers their computing and energy capabilities. By delegating its computationally resource demanding tasks to a set of nodes, a constrained device could thus establish secure, end-to-end communication channels with remote peers instead of relying on inefficient or vulnerable lightweight alternatives that include static shared secrets or use of an intermediary security gateway.

The contributions of this paper are summarized as follows: First, an overall overview of key establishment schemes and protocols was carried out. Its results were confronted to a study of IoT characteristics and requirements. Accordingly, relevant key establishment protocols, belonging to the key transport and key agreement families were identified as the best candidates fitting the end-to-end security requirements of the IoT. However, when assessing these protocols in terms of energy efficiency, we have illustrated the heavy computational cost they require to run on constrained devices. A study of how to securely and efficiently design collaborative versions of these protocols was conducted. The proposed collaborative schemes were designed so as to fit within the scope of current key establishment protocols (similar syntax and authentication model). Finally, we provided a detailed performance evaluation from the points of view of energy consumption (evaluation of computation and communication energy costs) and security (threat assessment and security measures) that proves the pertinence of our proposed key establishment approach.

In Section 2, we review existing key establishment protocols and assess their adaptability to the IoT paradigm. Then, we identify the Internet Key Exchange (IKE) and the Transport Layer security (TLS) Handshake as the most relevant key establishment protocols for the IoT and conclude on their inadequacy to efficiently ensure end-to-end security associations involving highly resource-constrained nodes. Section 3 describes the proposed cooperative key transport and key agreement approaches for IoT nodes. Instantiations of the collaborative approach are therefore proposed as updated versions of IKE and TLS Handshake protocols. Two distribution techniques, simple partition and threshold distribution, are detailed for each key establishment scheme. Section 5 discusses the performances of the proposed techniques as compared to the basic protocols from the point of view of the energy consumption. A security threat analysis of the collaborative approach is presented in Section 6. Section 7 concludes this paper.

2. Related work

Key establishment protocols, also named key exchange protocols, are used to "provide shared secrets between two or more parties, typically for subsequent use as symmetric

keys for a variety of cryptographic purposes" [5]. These purposes include the use of symmetric ciphers and message authentication codes, which are in turn used as security primitives for enabling various security protocols such as source authentication, integrity protection or confidentiality.

2.1. Classification of key establishment protocols

Key establishment protocols can be classified according to three criteria: the key delivery scheme (key transport or key agreement), the underlying cryptographic primitive family (symmetric or asymmetric) and the authentication method. The number of involved peers[1] (two, peer-to-peer or three, server-assisted) is sometimes added to these criteria. These notions are discussed in what follows.

2.1.1. Preliminary

2.1.1.1. Key transport vs. key agreement. A two-party *key transport* protocol is a protocol that runs between two peers, in which one or more secret value(s) are generated at one or both peers and securely transferred to the other peer. The resulting key is obtained as a function of the transferred secret values and possibly of other parameters that may have been exchanged as part of key transport.

In a one-pass key transport, only one secret value is sent from one of the peers to the other. The established key may be either this secret value itself, or may be derived from it along with other parameters, such as nonces. In a two-pass key transport, both peers exchange secret values that are used as input for the key generation function. Note that it is generally not safe to let one partner entirely control the key value.

A variety of server-assisted key transport is the distribution of a session key from a central server (key distribution center) to two peers. This requires of course that the central server be able to perform the distribution in a secure manner, e.g., through pre-established secured channels to both peers. Another, less frequent, variety of server-assisted key transport consists for the server to let one peer generate the session key, obtain it from this peer, and retransmit it over another secure tunnel to the second peer. In this second variety the assisting server is called a key translation center.

A two-party *key agreement* protocol is a protocol that runs between two peers, in which the resulting key is derived at both peers from public information exchanged between the peers. While each peer's public information takes the form of an encrypted secret, the decrypting of this encrypted secret by either the recipient peer or by the originating peer itself is never required to derive the shared key.

The Diffie–Hellman (DH) protocol [6] is the best known and most widely used key agreement protocol. It requires that two peers A and B first agree on appropriate prime (p) and generator (g). Then, A and B choose secret values, respectively a and b, compute the corresponding public values, respectively $g^a \bmod p$ and $g^b \bmod p$, and exchange these public values with each other. The same Diffie–Hellman shared secret K is then obtained at A by computing $(g^b \bmod p)^a$ and at B by computing $(g^a \bmod p)^b$ (see Fig. 1).

An often claimed security property of the Diffie–Hellman protocol is the perfect forward secrecy. This property ensures that the established secret could not be retrieved even though all long-term secrets of both peers are divulged. In the base Diffie–Hellman protocol, a and b are random numbers that are dynamically chosen as part of the key management protocol and immediately erased from memory afterwards. They could therefore not be qualified as "long-term secrets", which ensures that the Diffie–Hellman protocol fulfills the perfect forward secrecy property. This should not be generalized to all key agreement protocols, though. Some key agreement protocols are based on key pre-distribution. For example, the variant of the Diffie–Hellman protocol used in the HIP-DEX key establishment communication protocol (reviewed in what follows) requires that the Diffie–Hellman secrets a and b be statically fixed and remain the same in all key establishment operations. This use of Diffie–Hellman leads to losing the perfect forward secrecy property that is generally associated with it.

2.1.1.2. Cryptographic primitives. Both key transport and key agreement exist in embodiments that rely either on symmetric or on asymmetric cryptography. These cryptographic primitives should not be confused with those of the authentication mechanisms that may be integrated with the key establishment protocol and that are the subject of the next subsection. Let us take again the example of the Diffie–Hellman key agreement protocol. Diffie–Hellman is based on asymmetric cryptography primitives (actually, most of key agreement protocols are). Yet Diffie–Hellman, natively unauthenticated and vulnerable to man-in-the-middle attacks, has to rely on authentication techniques, some of which can be based on symmetric techniques.

Considering only the key delivery scheme and the cryptographic primitive type, four cases are possible:

Key transport with symmetric cryptographic primitives. This category regroups algorithms in which two peers, already owning a shared key, derive another one. Such operation typically happens when a symmetric key has to be refreshed, or when an ephemeral secret (e.g., transient session key) has to be derived from a long-term one [8].

- Key transport with asymmetric cryptographic primitives. In this category are found various key establishment protocols ranging from simple one-pass encryption of a secret key with a public key to more complex X.509 keying protocols [9,10].
- Key agreement with symmetric cryptographic primitives. A corresponding protocol, Blom's scheme, is presented in [1]. Although interestingly dissociating the key agreement notion from the Diffie–Hellman protocol, one cannot but notice that such cryptographic protocols are not used by main (and even minor) communication protocols.

[1] A key establishment protocol runs between two or more parties. In this paper though, we focus on peer-to-peer (pairwise) key establishment and do not consider the joint setup of a group key between more than two parties.

- Key agreement with asymmetric cryptographic primitives. With rare exceptions, this category is composed of the Diffie–Hellman protocol and its variants [11].

2.1.1.3. Authentication method. Authentication for a pairwise key establishment protocol relates to the ability, for one or both nodes that undertake it, to bind the established key material with the identity of its peer. While it is generally a good thing to have a pairwise key establishment protocol authenticate both peers to each other, it is not always the case. Commonly, only one peer is authenticated to the other; the authentication of the other peer, if required, has then to be ensured by another mechanism, possibly at another layer.

Some establishment protocols natively provide authentication. This is the case, for example, of a one-pass key transport protocol wherein a session key k is sent from a node A to its peer B, encrypted with B's public key. This protocol achieves indeed more than confidential key delivery: it proves to A that a node knowing k must be identified as B, since only B is expected to have been able to decipher the message containing k.[2] On the other hand, as mentioned above, the Diffie–Hellman protocol does not natively provide authentication. The Diffie–Hellman public values have therefore to be authenticated at communication protocol level, as is done by the IKE protocol, which ensures through digital signatures or keyed hashes that their origins can be validated.

Like those of key establishment protocol, the cryptographic primitives that underlie the authentication method can be classified as symmetric vs. asymmetric techniques. With the objective of defining the best practices for an IoT key establishment protocol, it is worth, though, going beyond this distinction and considering the underlying identity models. The categories of authentication that can be distinguished are listed hereafter. For clarity reasons, this list is made simpler by assuming that mutual authentication is desired, and that both peers use the same authentication method to each other.

- Shared secret-based authentication. This is the classical symmetric authentication scheme wherein two parties are statically configured with, or otherwise acquire, a common shared secret mapped to their respective identities [9].
- Static public key authentication. In this asymmetric authentication scheme, the two parties are statically configured with their respective public keys, mapped to their respective identities. Proving the knowledge of the corresponding private key implicitly ensures ownership of the matching identity [10,11].
- Certificate-based authentication. This is a variant of the previous category, wherein the mapping of a public key to an identifier is not a static configuration parameter but is obtained in the form of a signed certificate. Certificate-based authentication requires that a third party, the certificate authority, be trusted by both authenticating peers [10,11].
- Cryptographically generated identifiers. This family of asymmetric techniques changes the implicit assumption that any kind of identifier can be authenticated, provided that it is securely bound to a public key. These techniques assume indeed that the authenticated identifier of a node is obtained from the node public key, e.g., in the form of a hash of this public key. Mechanisms are then defined in order to build protocol stack identifiers (typically, IPv6 addresses) from these cryptographically generated identifiers [12].
- Identity-based authentication. This last set of asymmetric techniques bases on the Identity Based Cryptography paradigm wherein, oppositely to the previous category, a nodes public key is derived from its identity (whatever the format of this identity). Like in all asymmetric techniques, a node proves its identity by providing a proof of knowledge of the corresponding private key [13].

2.1.2. Synthesis

This subsection intends to provide a global view of the existing key establishment protocols, in order to ease the identification among them of the best candidates for IoT key establishment. This synthetic global view is provided in the form of a table, on which we chose to superpose the most known/used communication protocols. Usability of key establishment protocols currently in use in today's Internet is indeed a criteria that should not be left apart: the Internet of Things will definitely not start with a "clean slate" design approach, but will likely have to interoperate with widely adopted protocols of legacy Internet.

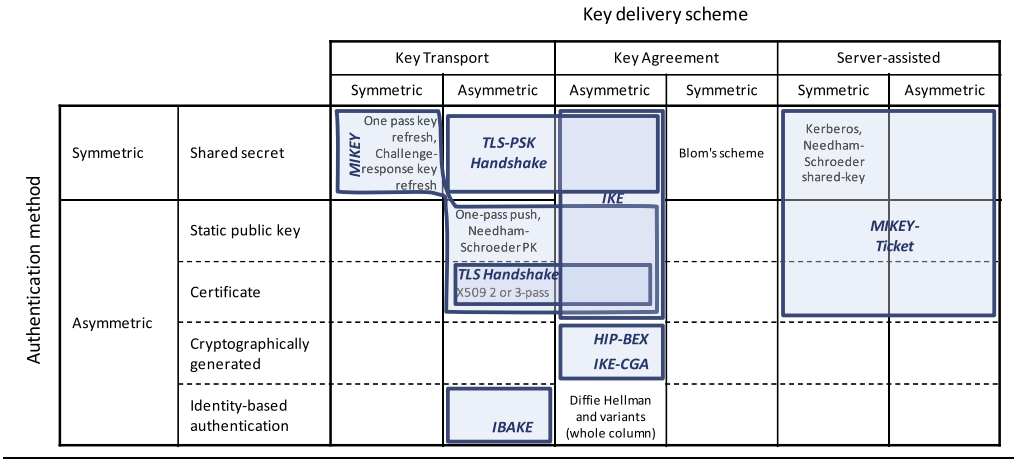
As can be seen in Table 1, the existing key establishment communication protocols mainly base on asymmetric cryptography, be it for the delivery/agreement scheme itself, or for the authentication method implemented within the protocol. Empty cells in the table are mostly found in the symmetric key transport and symmetric key agreement columns. Symmetric key transport protocols do exist, though; however, they essentially consist in key refresh/key derivation protocols, which we found did not fully qualify as key establishment protocols. Only the MIKEY protocol [8] is included in the column, since it is generally used to distribute session keys from long-term shared keys. Symmetric key agreement protocols are uncommon and require complex setup (pre-distribution).

2.2. IoT key establishment: specific requirements

This section reviews specific requirements that are involved in the identification of a key establishment protocol for the Internet of Things. These requirements fall into four main categories: those that are related to the fulfillment of security requirements, those that are related to pervasiveness (the Internet of Things is to encompass a wide variety of devices and networks, including legacy Internet), those that are related to efficiency (among IoT devices, most of them are resource-constrained) and those that are related

[2] The two steps of ensuring that only B may know the key k and obtaining the proof that some node knows the key k are respectively designated in [5] as *implicit key authentication* and *key confirmation*. Together, they form the *explicit key authentication* property.

Table 1
Classification of key establishment protocols according to the key delivery scheme and authentication method, with main key establishment communication protocols represented in overlay.



to adoptability or interoperability (the IoT should preferably use proven and deployed technologies and protocols).

2.2.1. Security

Contrary to wireless sensor network security, security in the IoT context involves end-to-end communications. The decentralized and bidirectional IoT communication paradigm also rules out the definition of static *client* and *server* roles: depending on the context, it is expectable that an IoT node will act alternatively as a client and as a server. These considerations translate into two security requirements. On one hand, end-to-end security should be provided. This means that only the two participants involved in the pairwise key exchange protocol should have access to the generated key. On the other hand, mutual authentication has to be provided. The two peers that establish a key between them should in the meantime authenticate to each other and bind the generated key to their respective identities.

2.2.2. Pervasiveness

By qualifying the Internet of Things as "pervasive", we refer to its foreseen universality, as a communication network interconnecting much more nodes than today's Internet, and actually encompassing this today's Internet. Pervasiveness puts additional requirements on a key establishment protocol for the IoT. Especially, it makes it highly unlikely that two nodes wishing to generate a key between them can leverage on a pre-existing security relationship based on long-term shared secrets or static public keys. For this reason, dynamic asymmetric key delivery schemes and authentication methods should be favored when designing an IoT key establishment protocol.

Pervasiveness also means that any two nodes may have to interoperate with each other, without considering their respective nature. Special care should therefore be taken, when designing an IoT key establishment protocol, to make sure that two nodes with important differences in capabilities be nevertheless able to generate a key with each other.

2.2.3. Efficiency

Efficiency has always to be considered when designing a new protocol. Four criteria are especially relevant when assessing cryptographic protocol efficiency: the number of exchanged messages, the bandwidth that is needed, the complexity of computations, and the possibility of pre-computations. Importance of these criteria increases when designing a protocol that will have to be run by highly resource-constrained nodes with low computational power, low memory, and limited battery capacity. Overall energy consumption, induced by both computations and message exchanges, is a good metric for these nodes. A protocol will be defined as more efficient than another if it obtains a metric value inferior to that of the other, while providing the same security level.

2.2.4. Adoptability

The Internet of Things will not emerge through the definition of entirely novel protocols. The approaches that rely on key generation schemes or authentication methods of limited usage should not be favored. Of course, interoperability mechanisms with these latter should be developed when desirable, though.

At this stage, it is worth quickly describing the two most widely adopted end-to-end security protocols we refer to in Table 1.

The Internet Protocol security (IPsec) [7] resides at the Network Layer of the OSI Model, which enables it to function independently of any application. It creates a secure (encrypted and/or integrity-protected) and authenticated tunnel between two endpoints, through which data can be exchanged safely, without being vulnerable to eavesdropping, packet forging/replaying or sender spoofing attacks.

Transport Layer Security TLS [10] provides the same security services at the transport layer while still being

application-independent. Hence, it can encapsulate higher-level protocols layering on top of the transport layer protocols. TLS has been designed to work with reliable transport protocols providing in-sequence delivery, such as the Transmission Control Protocol (TCP). Recently, a datagram-oriented variant DTLS has been proposed to operate on top of datagram-oriented transport protocols, such as the User Datagram Protocol (UDP). Both IPsec and TLS have the same design and provide equivalent security measures.

IPsec and TLS communication protocols rely on the use of cryptographic mechanisms such as encryption/decryption block ciphers and hash functions, in order to ensure the required security services for a communication. In turn, each of these mechanisms requires an initial key establishment phase allowing two communicating entities to authenticate each other and set up the required cryptographic keys. TLS protocol is preceded by a handshake protocol called TLS Handshake, which is responsible for key establishment and authentication. Likewise, the Internet Key Exchange (IKE) [11] is the most widely used IPsec keying protocol.

Each of these key exchange schemes independently implements specific techniques and cryptographic algorithms to derive a secret key and ensure the required mutual authentication between the endpoints of a communication.

2.3. Synthesis

From the requirements reviewed above, we can adapt our initial classification of key establishment protocols in order to identify among them the most suitable to the Internet of Things. The results of this identification are presented in Table 2.

Table 2 was obtained as follows. First, solutions relying on key pre-distribution were discarded, as they did not meet end-to-end security requirement. Then, solutions relying on symmetric cryptography or assuming initial knowledge of peer public key were discarded as they did not meet the pervasiveness requirement. It has to be noted that these first two requirements do not contradict each other: dynamic obtaining of asymmetric public keys through certificate and induced reliance on a certificate authority are different from letting the trusted third party generate the keys for both peers, and be thus in position to launch a key escrow attack. Finally, we disregarded in this work solutions that base on identity-based cryptography, which we considered not adopted enough.

The most relevant communication key establishment protocol candidates, retained from the juxtaposition of Tables 1 and 2, are summarized in Table 3.

2.4. Applicability of existing key exchange schemes for IoT scenarios and related work

Key establishment schemes used by IPsec and TLS as described above involve heavy public key cryptographic primitives and impact both energy and storage resources of a communicating entity. The resource constraints of most IoT components limit the implementation of these complex cryptographic mechanisms required to perform the key establishment, which could rapidly drain their resources and reduce the network performance. Existing end-to-end security protocols with their actual resource intensive design could not directly cope with the envisioned scenarios and requirements in the IoT. The feasibility of these security standards has to be revisited to adapt them to the IoT scenarios.

In the literature, several key establishment approaches have been proposed for traditional WSNs in order to cope with the resource constraint nature of sensor devices. Most of the proposed approaches rely on symmetric cryptography primitives due to their low resource consumption. Such solutions [14–17] based on pre-shared keys, are considered more efficient for sensor nodes. However, symmetric key based schemes are not targeting a secure end-to-end communication between the sensor node and remote

Table 2
Refinement of the key establishment protocols classification. Some candidates are ruled out either because they are not judged secure enough (no end-to-end security), or because they would not meet the IoT pervasiveness requirement, or because their adoptability is evaluated as low with respect to their use as of today. Efficiency is not discussed here, but will be the most important evaluation metric in the next section.

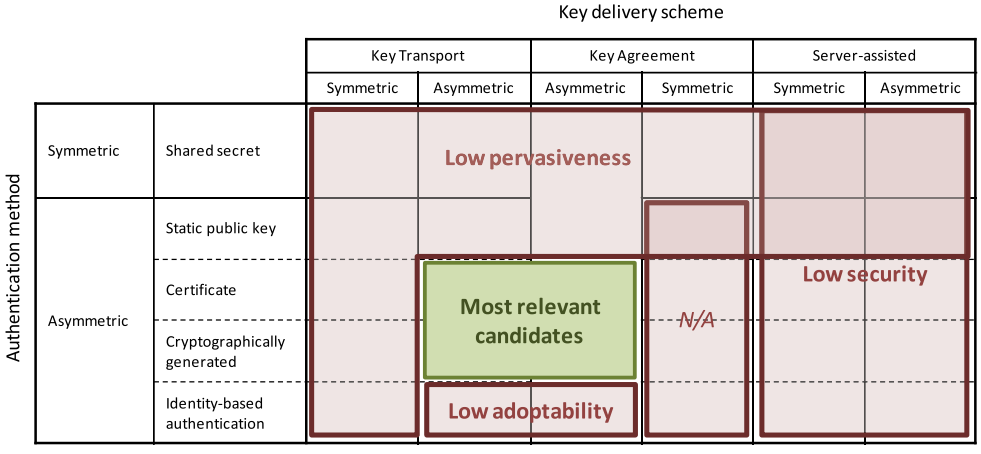


Table 3
Retained key establishment protocols for the IoT before considering the efficiency metric.

Protocol	Properties			
	Security protocol	Cryptographic protocol	Key delivery scheme	Authentication method
TLS Handshake	TLS	One-pass push/Diffie–Hellman	One-pass key transport/key agreement	Certificates
Internet Key Exchange (IKE)	IPsec	Diffie–Hellman	Key agreement	Certificates[a]

[a] Shared secret or static public keys IKE authentication methods are not considered here, since they do not meet the pervasiveness requirement. EAP-based IKEv2 authentication would likely rely on a certificate-based EAP method.

hosts. Instead, they offer security of communications within the sensor network wherein sensor nodes are connected to the internet via dedicated gateways. IoT requirements go far beyond those of WSNs since a sensor node in the IoT is considered as a part of the Internet able to establish end-to-end communications with external entities without requiring any initial knowledge of each other or any pre-shared keys.

Very recently, with the advent of WSN integration to the Internet, the need for an end-to-end security protocol between sensor nodes and the legacy Internet has been recognized. In order to enable functional implementations of TLS and IPsec in a constrained environment, lightweight variants of these protocols have been proposed. They base on the use of modified implementations of the corresponding keying protocols: TLS Handshake and IKE.

We have identified two different lightweight implementations of TLS Handshake on constrained devices that proposed to replace the heavy cryptographic operations of RSA and Diffie–Hellman algorithms with the use of Elliptic Curve Cryptography (ECC) during the key exchange. ECC is considered in the research community to be the most efficient algorithm among public key cryptosystems due to its lower energy consumption, fast processing time, compact signatures, and small key size [18]. Sizzle [19] was the first security protocol that proposed the use of TLS in the WSN in order to implement an HTTPS stack by enabling the use of ECC algorithms. Sizzle relies on translating gateways that map the sensor nodes local (non-IP) addresses to internet hosts IP addresses, allowing them to exchange data directly with remote IP peers. During the TLS Handshake, the Elliptic Curve Diffie–Hellman (ECDH) key agreement [20] and the Elliptic Curve Digital Signature Algorithm (ECDSA) [21], respectively replace the Diffie–Hellman key agreement and DSA algorithms. Favoring the use of these ECC-based protocols, performance evaluations showed that implementing HTTPS web servers on sensor nodes could be supportable for infrequent connections. SSNAIL [22] has been developed as a second lightweight TLS implementation for IP-based WSNs relying on the same cryptographic primitives as Sizzle for the key exchange while eliminating the use of the gateway. Authors measure that implementing an ECC-based TLS Handshake takes around one second while it takes 8.5 s for an RSA-based one.

Likewise, two recent variants of IKE have been proposed for energy efficiency purposes. Nagalakshmi et al. [23] modifies the IKE protocol by eliminating pseudo-random generation functions, thus eliminating its repetitive usage during the key exchange. The sender transmits a hash of its private key and its Diffie–Hellman private value instead of sending nonces. The proposed work leads to cost efficiency, however, the energy cost of a pseudo-random function generation (amounting to a symmetric encryption) can be neglected compared with the heavy cost of asymmetric cryptographic operations that are required further in the protocol exchange. In [24], an ECC-based IKE protocol has been proposed. It aims to reduce the heavy burden of the base exchange of the protocol IKE by using ECDH key exchange to set up the shared key and using ECC-based public key certificates for the authentication of the communicating entities.

However, recent studies have proved that the energy costs of ECC, being in the order of magnitude of millijoules, are still non-negligible when implemented on highly-constrained devices. Authors in [38] investigate the practical use of ECC on constrained devices and conduct a cost comparison between two key establishment schemes ECDH–ECDSA and Kerberos (a server-assisted key distribution protocol based on symmetric cryptography). They conclude that Kerberos is 95 times less costly than ECDH–ECDSA on a MICAz sensing platform. On the other hand, Liu and Ning [25] implemented ECC for MICAz and TelosB sensor platforms. They assessed ECC (160-bit keys) point multiplications cost needed to perform the ECDH exchange and ECDSA signatures. Results have shown that the energy cost of ECDH–ECDSA key agreement protocol is around 236 mJ for MICAz and 72 mJ for TelosB.

This unsuitability of prior key establishment proposals for constrained devices accentuates the need for novel IoT-specific solutions. This need was recognized in [26] and left open. According to the literature, the design of an efficient key establishment approach for existing end-to-end security standards that clearly addresses the heterogeneous IoT communications has not been undertaken yet [26]. Further careful design is required to reduce the energy cost of key establishment schemes while taking into account the heterogeneous nature and new communication paradigms of the IoT.

3. Proposed collaborative key establishment scheme

In this section, we tackle heterogeneity of IoT nodes from a different axis, trying to take advantage of it to design our solution for key establishment. We explore the possibility of reducing the computational load to be

performed on constrained devices during the key exchange instead of only thinking on reducing the cost of cryptographic algorithms (e.g., using ECC algorithms instead of RSA and DH algorithms), as proposed before. Eventually, we propose to exploit heterogeneity of nodes in order to offload heavy computational operations required at the constrained device to less constrained nodes in neighborhood.

During the key exchange, these assisting nodes, or "proxies", take charge of the session key derivation, in a collaborative and distributed manner. However, the session key is known only by the two endpoints of the communication, in order to guarantee its secrecy. Several constraints have been considered in the design of our approach: (i) the collaborative scheme must not come at the expense of a key disclosure risk or a collusion attack (ii) in case of a proxy unavailability or a greedy behavior, the system should continue to run properly (iii) each proxy is required to prove its legitimacy by proving that it is authorized by the constrained node to act on its behalf. Several constraints have been considered in the design of our approach (i) the collaborative scheme must not come at the expense of a key disclosure risk or a collusion attack (ii) in case of a proxy unavailability or a greedy behavior, the system should continue to run properly (iii) each proxy is required to prove its legitimacy by proving that it is authorized by the constrained node to act on its behalf.

3.1. Preliminary

3.1.1. Considered network model

Our network model considers a global IoT infrastructure that interconnects heterogeneous nodes with different capabilities in terms of computing power and energy resources. We especially consider in this work three different types:

- Highly resource-constrained nodes, unable to support the computation cost of asymmetric cryptographic operations required by the key exchange phase while requiring end-to-end security (e.g., sensor nodes).
- Proxies at neighborhood, less constrained and therefore able to perform cryptographic operations. These nodes may either be dedicated assisting servers or nodes belonging to the same local infrastructure employed for other applications, though being less impacted by energy constraints (e.g., having harvesting capability or line-powered devices).
- Unconstrained nodes not belonging to the same local infrastructure with high energy, computing power and storage capabilities (e.g., remote servers).

The considered scenario in this paper can be summarized as follows: a highly resource-constrained sensor node (the source node A) needs to exchange sensitive data with an external server (the destination node B) on an end-to-end basis. These two entities are supposed to have no prior shared key. Initially, their objective is therefore to setup a session key with each other. This scenario is likely to occur if one considers an IP sensor node (e.g., IPv6 over Low power Wireless Personal Area Network (6LoWPAN) sensor node) that has to deliver sensitive sensed data to remote peers with which it has not yet established shared secrets. This delivery may either happen through a *pull* model, wherein the sensor (IoT resource) is explicitly requested to provide data by a remote IoT requester, or through a *push* model, wherein the sensor is intermittently sleeping and regularly wakes up in order to push sensed data towards a (configurable) set of peers.

To illustrate the above scenario, let us consider the following example: a large office building connecting line-powered computers and other devices with different hardware configurations. The building is located in an earthquake prone area, hence, structural monitoring deployed by the government is used to periodically track weak spots in the walls of all buildings of the region, such that early measures can be taken to avoid damage and its possible catastrophic effects. For this reason, IP sensor nodes measuring vibrations and moisture are placed at or inside the walls of the office building. These sensors respond to the queries of remote seismic stations by providing them their measurements and thus allowing them to detect early earthquakes and send advanced alerts to the inhabitants of the region.

Besides that, the office building is equipped by surveillance sensors to keep an eye on its employees. One or two sensor nodes are also placed in each office to monitor the climate – temperature and humidity.

Among all the equipment operating in the building (computers, smartphones, and sensor devices), the seismic sensors are the less powerful nodes and are located in the less accessible places (that is, changing batteries is hard). On the other hand, they are to handle a highly sensitive task and their communications have to be established on an end-to-end basis basing on resource consuming cryptographic operations.

3.1.2. Assumptions

- After the initialization phase, each sensor node shares pairwise keys with a subset of its one-hop neighbors. These keys may have been generated during a specific bootstrapping phase using a trusted key management server or through more subtle mechanisms such as transitive imprinting. In case of node mobility, the key management server may take in charge the neighbor discovery and the distribution of pairwise keys can be dynamically performed when a constrained node will need to collaborate with a set of neighbors.
- The highly resource-constrained node is able to identify a set of less resource-constrained nodes that are available for supporting heavy cryptographic operations on its behalf.
- There exists a local trusted entity within the sensor network that owns a shared secret with all nodes in the sensor network and a public/private key pair.
- The external server does not communicate with the sensor network trusted entity but are statically configured with or able to validate its public key.

3.2. Preparation of involved entities

As an initial phase, the resource constrained sensor node A carefully selects the $P_1,\ldots,P_n$ proxies that will assist its key exchange based on the reputation of the nodes in the network and their actual resource capabilities.

Our approach requires that these nodes will process messages on behalf of the resource-constrained node during the key exchange. Hence, authorization and authentication questions arise at the proxy side, since these nodes should be provided with a representativeness proof. This proof could be a certificate including the proxy's public key associated with the right "authority to sign on behalf of A", all of which are signed with the source's private key and delivered 'offline' to the proxy, regardless of the current exchange. However, the use of long-time authorization certificates can be diverted for malicious exploits.

Hence, the certificate should include other dynamic parameters added by the source node in order to restrict the ability of proxies to act on its behalf, such as the identity of the destination node, a session nonce, or an expiration date. In this case, the authorization proof should be delivered 'online' to the proxy during the protocol exchange. Nevertheless, managing dynamic certificates would be hindering for the constrained sensor node.

For this reason, we propose to move the computational load required to dynamically manage authorization proofs from the sensor node to a local trusted entity T, which will be the only entity able to assert that a proxy node is authorized to sign on behalf of A. On the other hand, the verification of each proxy's certificate would be also resource consuming for the destination node; we propose to rely on the technique of authenticated dictionaries such as Merkle tree [27] or one-way accumulators [28] to efficiently authenticate participants and validate their membership to the group of selected proxies at the server side. These techniques provide a means to authenticate a high number of items without individually validating each of them, but rather authenticating them as a whole. Indeed, the items to authenticate are placed in the leaves of a binary tree. The item corresponding to a parent node is computed from the items of its two children, e.g., through a one-way hash function. Eventually, all leaf items are involved in the computation of the root node value. Thus, only this value has to be authenticated in order to authenticate all items of leaves. The membership of a leaf in the group can be then verified with respect to a publicly known root value and its authentication path, this latter being defined as the successive items required to compute the root value from the considered leaf. Using this technique, the destination node has only to verify the signature of the root value (computed and signed by the local trusted entity) to authenticate all proxies' public keys.

Upon receiving their proof material, proxies are prepared to participate to the collaborative process.

3.3. Key exchange description

In order to give a clear description of the proposed collaborative process, we treat each class of key exchange apart.

3.3.1. Collaborative key transport

In this section, we describe how we offload the key transport computational load from a highly constrained node to a set of proxies. We consider first the one-pass key transport mode and then adapt the proposed solution to the two-pass key transport mode.

3.3.1.1. Collaborative one-pass key transport. During the one-pass key transport mode, the highly resource-constrained node A generates a random secret key x and relies on a set of proxies to deliver it to the server B using asymmetric cryptography. We propose two techniques to distribute computations required for the secret key delivery.

The successive phases that make up our proposal are illustrated in Fig. 2 below, and explained later in the following subsection.

a. *Simple secret partition*

A starts by splitting the secret x into n parts $x_1,\ldots,x_n$ and then sends each part x_i to the corresponding proxy P_i.

Upon reception of the x_i secret key part, the proxy P_i encrypts it using the server's public key. The result of encryption is then signed using the proxy's private key for message integrity and authenticity.

We propose to use the lightweight one-time signature scheme of Lamport [29] in order for the proxy to sign messages on behalf of the constrained node. This signature scheme is especially lightweight and computationally efficient compared to other signature schemes [30]. Two drawbacks could possibly mitigate its practical applicability: on one hand, a public/private key pair should be used only once since information about the private key is divulged along with the signature itself. On the other hand, a long key will be needed to sign a long message, since the private (resp. public) key is the concatenation of all private (resp. public) values, as numerous as the message blocks and being each as long as the associated hash function output. Nevertheless, neither of these shortcomings

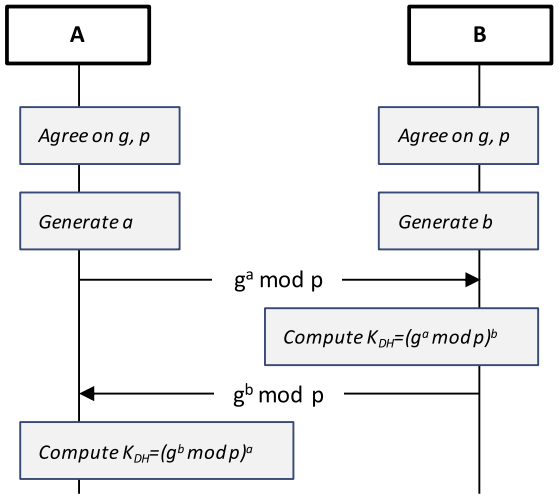


Fig. 1. Diffie–Hellman key agreement.

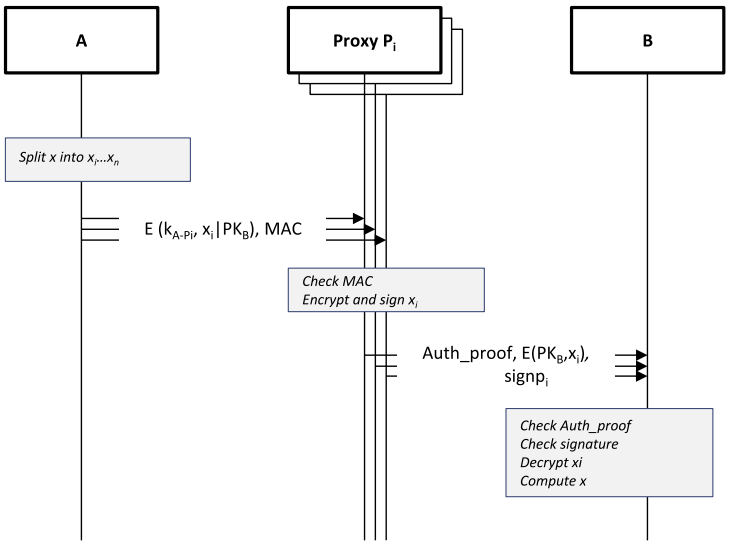


Fig. 2. Collaborative one-pass key transport.

affects our approach, which addresses one-time exchanges of short messages. In this case, we propose that T generates the Lamport private/public keys for each proxy P_i and securely provides it with this key material along with the authorization proof in the same message.

After receiving the required key materials, the proxy signs the encrypted secret key x_i and then sends the result to the server B. In turn, B verifies the integrity of the received message using P_i's public key and eventually decrypts x_i.

We assume that each proxy P_i has initially contacted B in order to request for its certificate and to provide it with its own certificate and its proof material. In response, B verifies that the proxy has supplied a valid public key along with information from T asserting that it is a valid proxy assisting A in its key establishment process.

Having received all x_i fragments, B becomes able to recover the original secret key x.

b. *Threshold secret distribution*

At this stage, it is worth noting that the proposed solution is based on reliable deliveries of all secret fragments x_i in order to be able to reconstitute the source's secret key at the destination node. A single missing message from a proxy makes the information incomplete for the server and may fail the protocol exchange.

Assuming that proxies behave as honest and reliable participants could be difficult in practice: even in scenarios where dedicated trustworthy proxies are made available to resource-constrained nodes, reliability of those proxies is not guaranteed. Hence, in case of unavailability or non-cooperative behavior of a proxy, a retransmission operation, optionally preceded by a new proxy assignment may have to take place. However, the proposed system will suffer from an additional latency.

In order to reinforce the reliability of the proposed distributed scheme, a forward error correction scheme [31] will be applied by the source A to handle losses and missing secret parts from assisting nodes.

The principle of forward error correction scheme is to add redundant parity packets to the original message, divided into multiple packets, in order for it to be recovered by the receiver even if some packets were altered or lost during the process of transmission. Let n be the total number of sent blocks, k ($k < n$) is the minimum number of blocks required to reconstruct the original message. First, the source node performs the split process of the secret key. Then it applies the error redundancy scheme to the fragments of the secret key as depicted in Fig. 3 below.

Then, the server B becomes able to reconstruct the session key provided that a sufficient number of packets from assisting nodes are received, without requiring the reception of *all* of them. This technique protects our solution from unreliable delivery in proxy → server connection, though the source node should perform more computational operations in the initial phase, to add redundancy to the secret key.

3.3.1.2. Collaborative two-pass key transport. In two-pass key transport mode, a random secret key x generated by

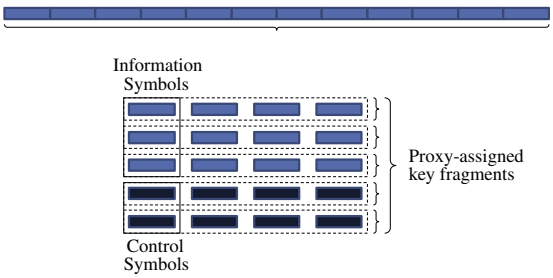


Fig. 3. Adding redundancy for reliable one-pass key transport.

the constrained source A and a second random secret value y generated by the server B are used to compute the session key. The phases of the proposed solution are depicted in Fig. 4 below.

We propose to apply the same collaborative approach as described in the one key transport scheme to deliver the secret x from the source to the server. After having received a sufficient number of x_i fragments, the server obtains the secret value x. At this stage, it generates a secret key y to be provided to the resource-constrained client. However, this latter cannot decrypt and verify the authentication and the integrity of the received value because of its resource constraints. For this purpose, we propose that the proxies also support the reception of the secret key y on behalf of A in a cooperative manner. These nodes take charge of the computational load required to verify the received message from the server and then transmit it securely to the source. Yet, the divulgation of the secret key y to the proxies would affect the security of our system. In order to preserve the secrecy of y, we propose to have it encrypted with the secret key x reassembled at the server. The x-encrypted secret key y is MACed with the secret x and then signed with the server's private key. It is finally sent to each proxy P_i, which has to verify the integrity of the received packet from the server before decrypting it. Then the packet content (that is, y encrypted and MACed with x) is securely transmitted to the client. As long as an appropriate number of the same packet is received from different proxies, the client ensures the validity of the transmitted message from the server. Consecutively, it checks the MAC in order to ensure that the server has obtained the same secret x and verify the message integrity. Once the client A receives a valid message, it can obtain the transmitted secret value y in order to complete the set-up of the session key.

3.3.2. Collaborative key agreement

The key agreement process involves heavy cryptographic computations on both parties. The most requiring part is the computation of two modular exponentiations, necessary for the generation of the Diffie–Hellman public keys and the setup of the Diffie–Hellman shared key. Applying the same collaborative approach in the above section, we propose to delegate the heavy cryptographic load to less constrained nodes in neighborhood. In the two following subsections, we describe two techniques that we introduce in order to distribute the computations required by the Diffie–Hellman protocol and therefore to enable the key agreement protocol. For each of these techniques, we explain how the source's DH private key is shared among proxies (how A computes a_i), how the server retrieves the source's DH public key from the proxies' g^{ai} mod p, how the server computes the shares B_i of its own DH public key (how B computes B_i) and how the proxies use B_i to obtain the K_i shares of the DH session key K_{DH}, eventually used by A to retrieve K_{DH}. The collaborative protocol exchanges are illustrated in Fig. 5 below, and detailed later in this subsection.

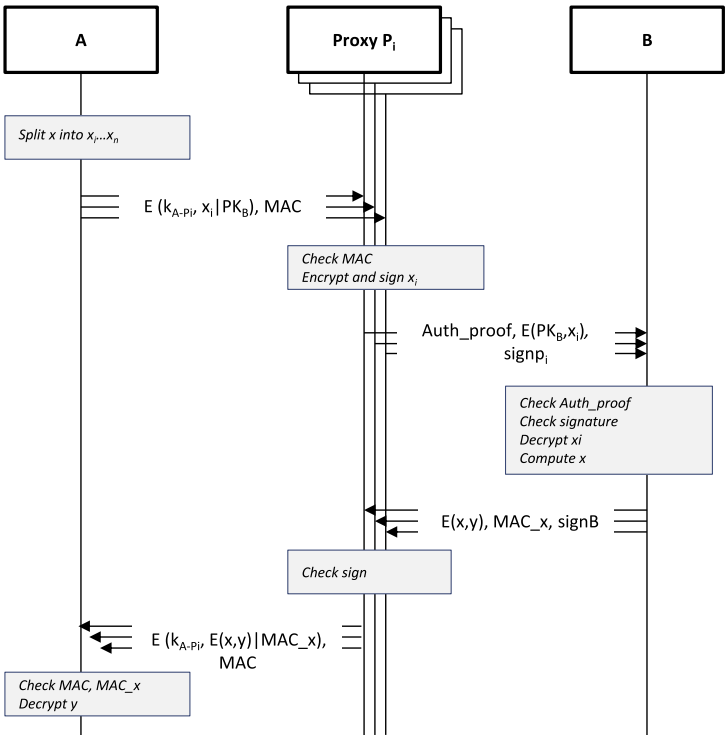


Fig. 4. Collaborative two-pass key transport.

3.3.2.1. Secret exponent integer partition. The technique described in this subsection is the simplest approach for enabling distributed DH key exchange. The secret exponent a of the source is split into n parts $a_1,\ldots,a_n$ chosen such that:

$$\sum_{1}^{n} a_i = a \bmod p \tag{1}$$

Upon reception of a_i, each proxy P_i computes its part of the initiator's DH public key $g^{ai} \bmod p$ and delivers it (signed) to the server. The computation of the source's DH public key eventually occurs at the server and amounts to the product of the values received from the proxies, following:

$$\prod_{i=1}^{n} g^{a_i} \bmod p = g^{\sum_{i=1}^{n} a_i} \bmod p = g^{a} \bmod p \tag{2}$$

In turn, the server sends a share B_i of its DH public key to each proxy P_i. With this first simple partition technique, B_i is equal to the server's DH public key for each proxy. The computation by each proxy of the share K_i of the DH session key occurs then as follows:

$$K_i = B_i^{a_i} = (g^b \bmod p)^{a_i} = g^{b.a_i} \bmod p \tag{3}$$

Eventually, the computation of the DH session key is made by the source, which obtains K_{DH} as:

$$K_{DH} = \prod_{i=1}^{n} K_i = \prod_{i=1}^{n} g^{b.a_i} \bmod p = g^{b.a} \bmod p \tag{4}$$

According to this expression, the resource-constrained node only spends $n-1$ modular multiplication operations instead of two modular exponentiation operations, with exponents of considerable length (a and b should have twice the length of the generated secret K_{DH}, as per [32]).

3.3.2.2. Secret exponent threshold distribution. The previous solution is based on reliable multiple hop-by-hop deliveries of secret fragments, each fragment a_i being the ith summand of a modular integer partition of the source's DH private key. The server needs therefore to receive all messages from all proxies in order to be able to reconstitute the source's public key. A single missing message from a proxy makes the information incomplete for the server and may block the protocol exchange.

In order to reinforce the reliability of the proposed distributed scheme, this kind of defective proxy play has been carefully considered in the design of this second proposed approach for key agreement. We have implemented a robust technique that ensures a consistent recovery of the source's DH public key at the server even in case of a proxy misbehaving or unreliability. Note that the technique introduced above for key transport could not be adapted to a key agreement protocol, where the secret exponent a is never retrieved at B's side.

The enhanced distributed approach we propose is based on the use of a (k, n) threshold scheme, wherein the n proxies obtain a polynomial share instead of a partition element, k polynomial shares being enough to reconstruct the source's DH public key through the technique of Lagrange polynomial interpolation. In cryptography, Lagrange polynomials were initially used in Shamir's secret

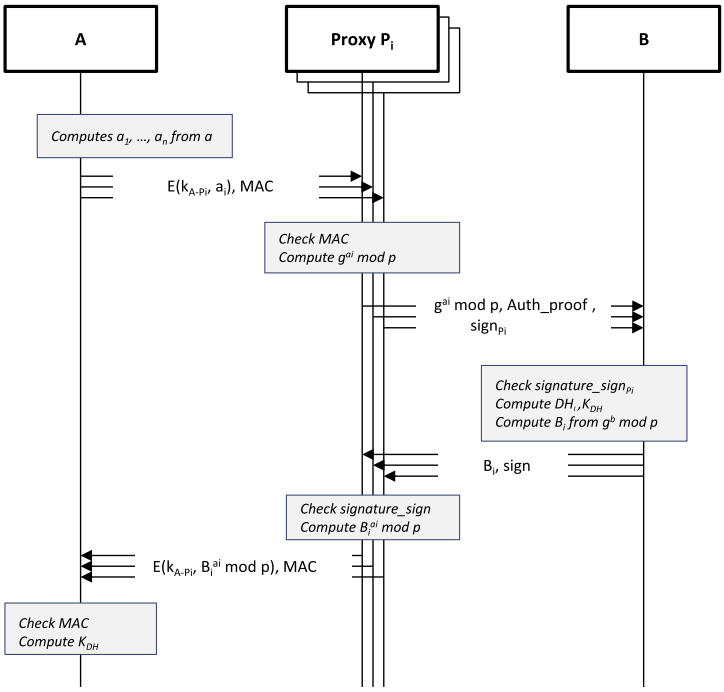


Fig. 5. Collaborative key agreement.

sharing schemes [33]. The proposed threshold scheme satisfies the two properties that the integer partition solution fails to provide:

(1) *Recovery:* The server can recover the source's public key provided that a sufficient k values from proxies are received, without requiring the reception of all of them.
(2) *Secrecy:* Nothing is learned about the secret exponent a even if $k-1$ shares of it are disclosed. In other words, data delivered to the server through proxies in order to compute the source's public key will not reveal partial information about secret exponent.

Given a polynomial function f of degree $k-1$ expressed as: $f(x) = q_0 + q_1 x + \cdots + q_{k-1}x^{k-1}$ with $q_1, q_2, \ldots, q_{k-1}$ being random, uniform and independent coefficients and $q_0 = a$.

Applying the Lagrange formula, the polynomial f can be retrieved as follows:

$$f(x) = \sum_{i=1}^{k}\left(f(i) \times \prod_{j=1, j\neq i}^{k} \frac{x-j}{i-j}\right) \tag{5}$$

From (5), the secret exponent a can be computed given any subset of k values of $f(x)$:

$$a = f(0) = \sum_{i=1}^{k}\left(f(i) \times \prod_{j=1, j\neq i}^{k} \frac{-j}{i-j}\right) \tag{6}$$

In our threshold distributed approach, the distributed shares a_i of the private exponent a are obtained as $a_i = f(i)$. So, in order to bootstrap the threshold distributed key agreement, the source calculates n values $f(1), \ldots, f(n)$ of the polynomial f, with $n > k$, and sends each $f(i)$ to the correspondent proxy P_i. Each proxy computes then its part of the source's DH public key $g^{ai} \bmod p = g^{f(i)} \bmod p$ and sends it to the server.

Upon the reception of a subset P of k values transmitted by the proxies, the server starts by computing the c_i coefficients as follows:

$$c_i = \prod_{i\in P, j\neq i} \frac{-j}{i-j} \tag{7}$$

Then, B computes the source's DH public key DH_I based on the Lagrange formula:

$$\begin{aligned}\prod_{i\in P}\left(g^{f(i)}\right)^{c_i} \bmod p &= g^{\Sigma_{i\in P} f(i)\times c_i} \bmod p \\ &= g^{f(0)} \bmod p \\ &= g^{a} \bmod p\end{aligned} \tag{8}$$

In order to prepare the computation of the DH session key at the source side, B starts calculating for each proxy P_i ($i \in P$) the value $B_i = g^{b.ci} \bmod p$ (c_i being the ith coefficient calculated in the previous phase). P_i is unable to compute the coefficient c_i since it has no knowledge about the subset P of concrete participating proxies. Having received this value, each proxy P_i uses its share $f(i)$ of the source's private exponent to compute $K_i = B_i^{f(i)} = g^{b.ci.f(i)}$. Each proxy delivers then this computed value to the source A.

Upon reception of these k values, the source computes the DH session key K_{DH} as follows:

$$\begin{aligned}K_{DH} &= \prod_{i\in P} g^{bf(i)c_i} \bmod p \\ &= g^{b\Sigma_{i\in P} f(i)c_i} \bmod p \\ &= g^{ab} \bmod p\end{aligned} \tag{9}$$

By applying the threshold technique to improve the effectiveness of the distributed approach, the source is led to perform more computational operations in the initial phase, in order to calculate the n values of the polynomial that it sends to the n proxies. The cost of the computation can be better estimated if one considers another way of writing $f(x)$, as:

$$f(x) = (\cdots((q_{k-1}x + q_{k-2})\cdot x + q_{k-3})x + \cdots)\cdot x + q_0 \tag{10}$$

According to this expression, A performs for each computation of $f(i)$: $(k-1)$ multiplications between a scalar and a large number and $(k-1)$ summations of two large numbers. It is worth noting that k and n are small numbers, smaller than the number of secure relationships that the source is able to maintain. On the other hand, the polynomial coefficients are as large as the DH private key of the source.

4. Collaborative IoT key establishment protocols

We show in this section how our proposed collaborative approach, under its integer partition and threshold distribution embodiments for key transport and key agreement modes, can be applied to the IoT key establishment protocols (TLS Handshake and IKE) that were identified in Table 3 of Section 2.

4.1. Modified TLS Handshake

4.1.1. Basic TLS Handshake

When a TLS connection is needed between a client and a server, an initial phase called TLS Handshake [8] is needed to negotiate security algorithms, to authenticate at least one peer to the other and to establish a shared secret between both peers. The TLS Handshake protocol supports two different key exchange methods. We consider in this paper the key transport method based on RSA asymmetric cryptography. The entire exchange is illustrated in Fig. 6.

First the client and the server exchange Hello messages. These messages contain nonces and negotiate the set of cryptographic algorithms that will be applied to the session. The server Hello also contains the server's certificate and a signature for authentication.

Next, the client sends a message containing a generated secret – called premaster key (PMK). In the key transport mode, TLS Handshake performs the one-pass key transport so that the premaster key is pushed from the client to the server. The PMK is thus encrypted using the server's RSA public key, which is retrieved from the server's certificate. This message also includes a signature on the hash value of the PMK, combined with all past messages exchanged during the current session. The client authentication is option-

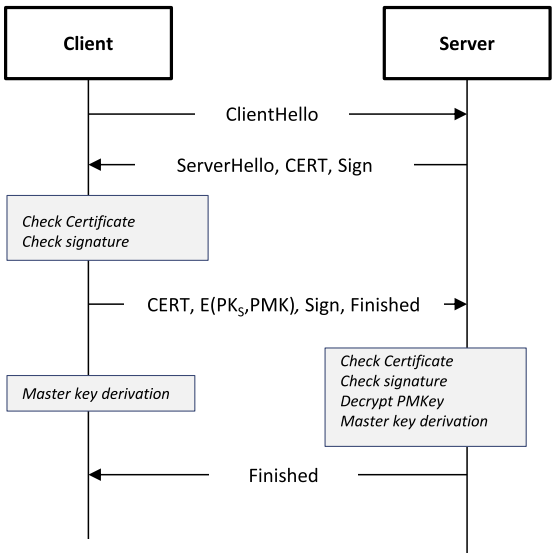


Fig. 6. Basic TLS Handshake (key transport mode).

ally performed during TLS Handshake: if requested by the server, the client provides it with a certificate and a signed message.

In order to reduce the PMK storage requirement at the communicating parties and to ensure the key freshness, a master secret is derived from PMK using a hash function applied to the concatenation of the PMK and the two nonces exchanged in Hello messages.

The Finished message ends the handshake exchange. It includes a hash computed over the master key and all the past messages. The receiving entity is able to compute the corresponding hash value from its own records in order to check if the result matches the received value.

4.1.2. Collaborative TLS Handshake in the key transport mode

The protocol exchange is illustrated in Fig. 7 below and detailed afterwards. Message exchanges are alike when considering either the threshold secret distribution or the simple secret partition technique. This is because the redundancy scheme is applied at the client before the delivery of the premaster key.

The Hello messages are similar to those of the basic TLS Handshake. As described before, both of these messages include random values used as nonces to prevent replay attacks and to compute the session key.

Upon successful connection with the server, the constrained client needs to verify the server certificate (using the Certificate Authority (CA) public key) and signature (using the server public key) and has to securely provide the server with a premaster secret *x*, used later to compute the shared master key. At this stage, it is worth noting that the verification operations, each performed with an RSA public key, can be supported by the constrained device since they are far less resource-demanding than signature

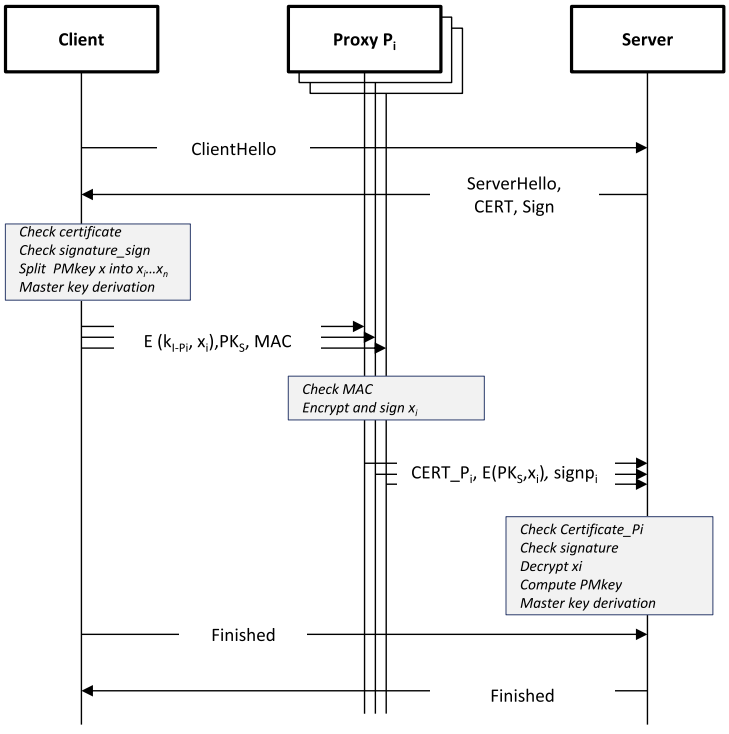


Fig. 7. Distributed TLS Handshake (key transport mode).

operations involving the use of a private key in RSA cryptosystems. Delegating these verification operations would be more resource demanding for the constrained node since it would have first to forward an around 1000 bytes certificate to each proxy.

Once it has verified the legitimacy of the server, the client calls on the proposed cooperative process. It first applies an error redundancy scheme (in case of a threshold secret distribution) to the original premaster key x, splitting it into n parts $x_1, \ldots, x_n$. It then sends each part x_i along with the server public key to the corresponding proxy Pi. At this stage, proxies take in charge the cooperative transmission of the premaster key as described above. The protocol exchange ends with two 'Finished' messages, exchanged between the server and the client, as in the TLS basic handshake, to ensure that the master key has been correctly recovered at both parties (*mutual key confirmation property*).

4.2. Modified Internet Key Exchange (IKE)

4.2.1. Basic IKE

The objective of IKE [9] is to establish a secure channel between two parties and enable them to mutually authenticate each other. IKE provides a protocol to establish security associations (SAs) that are needed to secure IP datagrams using IPsec. It only performs the key agreement mode:

All IKE communications are in the form of request–response pairs. An IKE transaction consists of two required request/response exchanges, as depicted in Fig. 8. The first request/response exchange (IKE_SA_INIT) negotiates cryptographic algorithms (SAi1, SAr1), exchanges nonces (Ni, Nr) and performs the Diffie–Hellman exchange to establish a shared key. The messages in this exchange are not authenticated; the following exchanges authenticate these messages by including their content while calculating the authentication values. At this stage, both sides have enough information to set up a master key K_M, using both Diffie–Hellman public values and the nonces. All shared keys for the IKE SA are then derived from this master key.

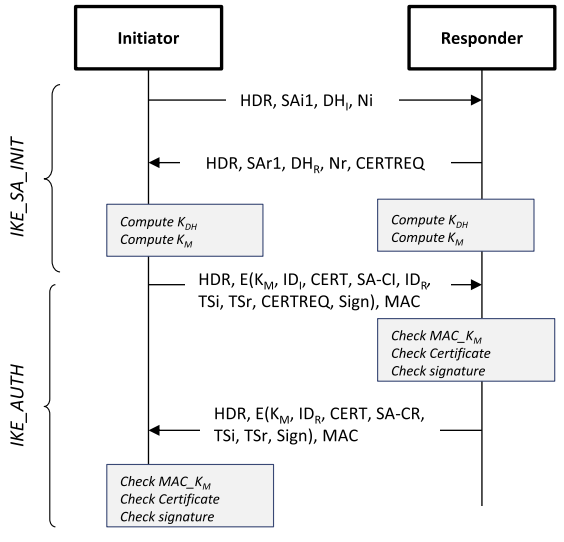


Fig. 8. Basic internet key exchange (establishment of a simple SA).

The second request/response exchange (IKE_AUTH) authenticates the previous messages. The identities of both sides are authenticated, and a simple IPsec SA, called a child SA, is established. Security association descriptions (SA-CI, SA-CR) indicating the supported cryptographic algorithms and the traffic selectors (TSi, TSr) are exchanged. Parts of these messages are encrypted and integrity protected (with a MAC) using the master key established in the IKE_SA_INIT exchange.

At this stage, the IKE transaction has been authenticated and a single child SA has been established. If no other child SAs are required, the IKE transaction terminates here. If, however, additional child SAs are required, the transaction moves to create another child SA.

4.2.2. Collaborative IKE

The figures below describe the modified protocol exchange obtained by applying the collaborative key agreement with the two proposed techniques.

As in the basic IKE, this modified variant also consists of two phases. During the IKE_SA_INIT phase, the two peers perform the Diffie–Hellman key agreement relying on the assistance of proxies as described above and finally derive a master key K_M using both the DH shared key and the nonces (Ni, Nr). During this phase, proxies also provide their certificates to the responder contrary to what happens in the basic protocol exchange. This makes the responder in a position to check the legitimacy of proxies acting on behalf of the initiator and to obtain the reconstitution parameters required to compute DH values. At this stage, proxies' messages are still not authenticated in order to keep authentication process for the second phase, as in the basic IKE.

During the IKE_AUTH phase, the initiator delegates the computational load of the signature and verification operations to the proxies in a distributed manner. It first exchanges with the responder encrypted messages using K_M for key confirmation indicating the supported cryptographic algorithms and the proposed traffic selectors (TSi, TSr). Then, it triggers the authentication process between the proxies and the server through the message 'AUTH_-start' as illustrated in the sequence exchanges below. Once both sides are authenticated, proxies provide the initiator with an 'AUTH_success' message ending the IKE_AUTH phase.

Figs. 9 and 10 below represent how the proposed approaches with simple integer partition (Fig. 9) and threshold distribution (Fig. 10) are used with the IKE protocol.

5. Performance analysis

As described above, our collaborative approach introduces a communication overhead due to the message exchanging between the source, the trusted entity T and the proxies. A performance analysis is therefore required to assess the respective efficiency of the proposed collaborative TLS Handshake and IKE and compare them with the basic approaches used for the key exchange.

5.1. Computational cost

In order to precisely quantify the energy savings at the constrained source node, we have implemented the cryptographic operations it performs in TLS Handshake and IKE protocols, considering both their basic and collaborative approaches. We have evaluated their cryptographic energy costs using Crypto++ library [34]. The number of proxies involved in the collaborative key establishment scheme is set to 5. With respect to error correction, we have chosen to rely on the Reed–Solomon (RS) code [35] in the threshold distributed approach of TLS Handshake in its key transport mode. In our simulation, we use RS (5, 3) ($n = 5$, $k = 3$) codes where we generate 2 parity packets for 3 source packets. The computational energy cost of RS code was evaluated using IT++ library [36].

Test programs for individual computational operations were run on an Intel i3 processor and the corresponding number of processor cycles for each was retrieved. In order to be able to induce the number of cycles measured on a resource-constrained device from the number of cycles on a powerful processor, we disabled advanced features of our test processor (hyperthreading, multi-core, variable clock speed). Eventually, we were able to consider that the number of cycles C_{TelosB} can be derived from the number of CPU cycles measured on the i3 (C_{i3}), under the following equation:

$$C_{\text{Telos B}} = \frac{\text{Register\_size}_{\text{i3}}}{\text{Register\_size}_{\text{TelosB}}} \cdot \alpha \cdot C_{\text{i3}} \tag{11}$$

where α is a coefficient representing the richer instructions of the i3 and approximated to 2 in our analysis.

The total energy cost of a specific operation for a sensor (E_{TelosB}, expressed in Joules) can be calculated by multiplying

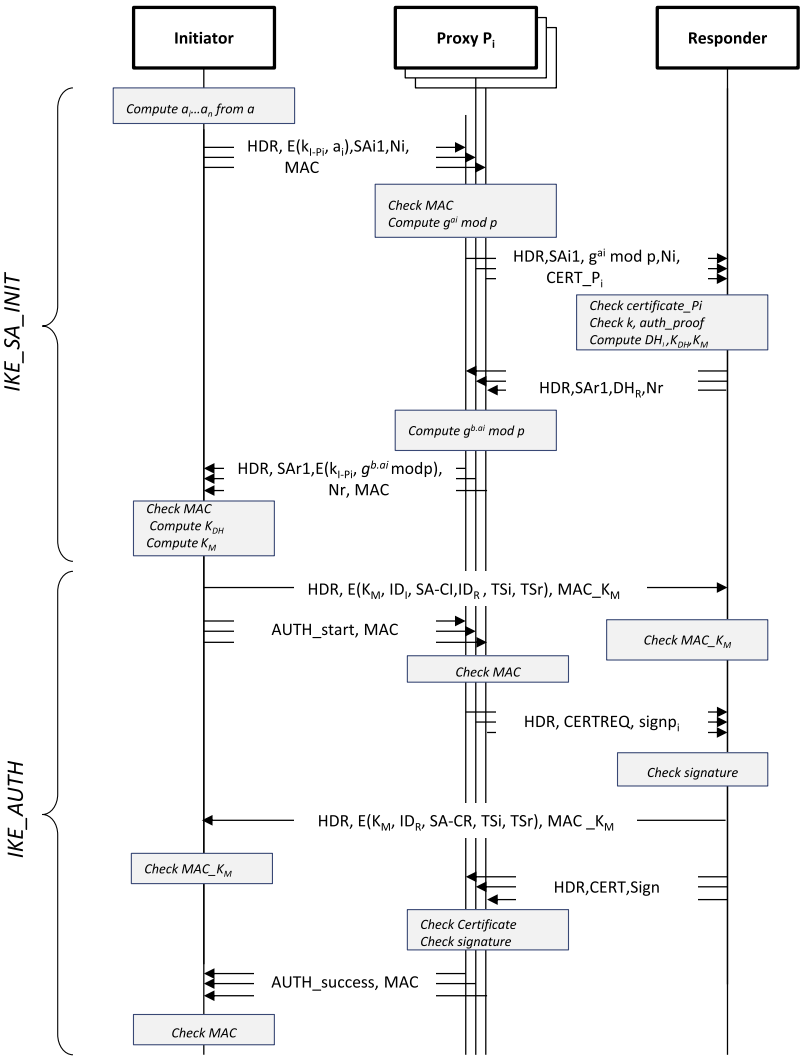


Fig. 9. Distributed IKE: simple integer partition technique.

the energy consumption per CPU cycle with the estimated number of CPU cycles (C_{TelosB}):

$$E_{TelosB} = \frac{U_{TelosB} \cdot I_{TelosB}}{N_{TelosB}} \cdot C_{TelosB} \quad (12)$$

where U, I and N are respectively the voltage, intensity and frequency of TelosB.

Computational cost results for distributed TLS Handshake (representative of a one-pass key transport protocol) and distributed IKE (representative of a key agreement protocol) are respectively presented in Table 4 below.

5.2. Communication cost

In this subsection we assess the communication energy costs of the proposed distributed approaches at the constrained initiator. These costs are made of the costs of transmission, reception and listening. The energy consumption of a node in listening mode can be equivalent to its consumption in reception mode since the transceiver remains active in both modes (see Table 5).

Authors in [37] assess the energy cost of cryptographic algorithms at resource-constrained nodes and reveal the impact of listening on the total energy cost. However, they did not consider this element in their estimates. [38] includes the listening cost to estimate the energy cost of ECDH–ECDSA and Kerberos protocols on TelosB and MICAz sensors and insists on its importance comparing results with a prior work that estimates communication cost considering only transmission and reception costs. This comparison shows an energy overhead of 45% when the listening cost is taken into account.

We use the power consumptions presented in Table 5 as an energy model of the different operating modes (transmit, receive and listen) for the TelosB platform [38]. As reported in [38] we consider an effective data rate of

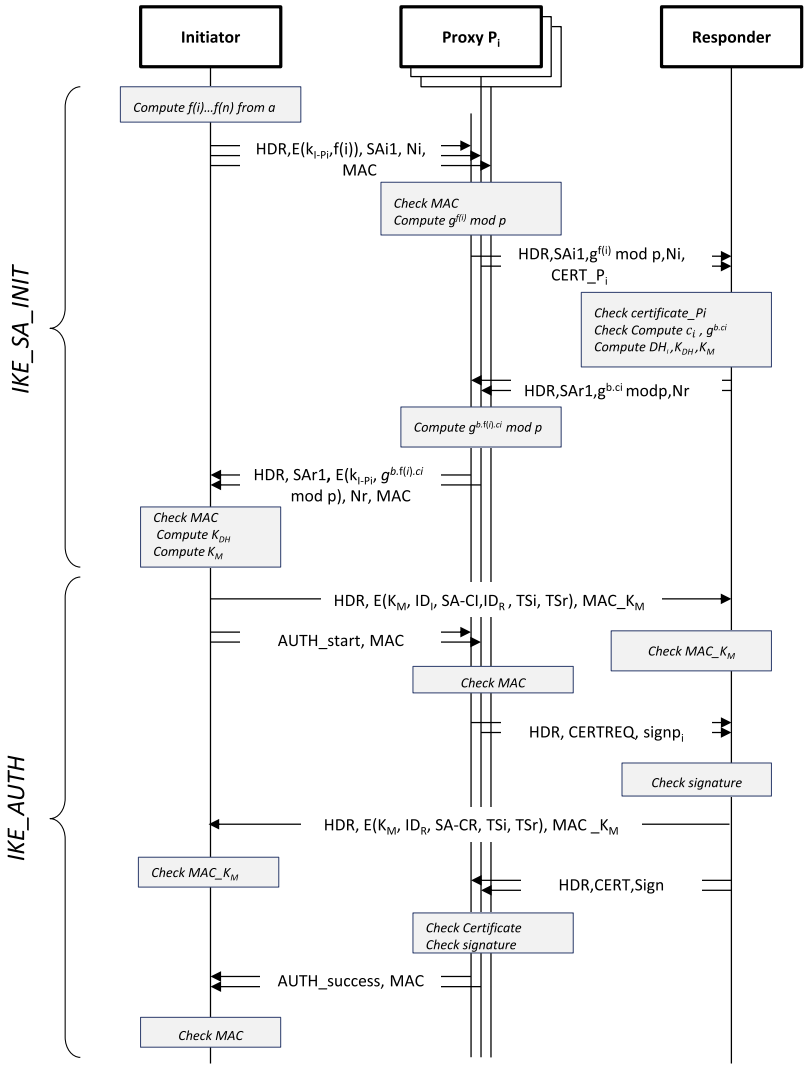


Fig. 10. Distributed IKE: threshold secret distribution technique.

75 kbps for a 250 kbps claimed one. This important decrease of the data rate is discussed in [39]. From the previous exchange descriptions, we obtain in Table 6 below the number of exchanged bytes by the source node in TLS Handshake and IKE protocols, considering both the basic exchange and the distributed approaches.

We consider that the constrained node is listening during a delay corresponding to the latency of communications (T_x, R_x) and packets propagation (Δ) as well as the processing of packets at the proxies and the responder. We estimate below the listening durations required by the constrained node in the considered approaches:

$$\Delta t_{Listen} = \underbrace{Proc(P_i) + \underbrace{T(P_i) + \Delta(P_i \rightarrow R) + Proc(R) + \Delta(R \rightarrow P_i) + R(P_i)}_{Basic\ Approac\ h\ listening\ time} + Proc(P_i)}_{Distribute\ d\ Approac\ h\ listening\ time}$$

Assuming that the server is an unconstrained node while proxies are 10 times less constrained than the server, this duration is respectively 401 ms and 404 ms for TLS Handshake and IKE while it respectively amounts to 411 ms and 446 ms for the distributed TLS Handshake IKE approaches. We also assume that the proxy is one hop far from the initiator and that a 200 ms propagation delay is required to route packets from the source to the server.

Finally, the energy costs induced by communications in both basic approach and distributed approaches is shown in Table 7.

5.3. Total energy cost

Gathering the computation and communication costs, we provide the total energy costs of the two examples of key exchange protocols considering the basic and collaborative approaches in Fig. 11 below.

As shown in Fig. 11, the computed costs confirm the efficiency of the cooperative scheme we propose. The most significant energy savings concern the key agreement mode. They amount to 80% of what is consumed in IKE protocol. Concerning the key transport of TLS handshake, the constrained node saves around 35% of its energy, as compared with what is spent during the basic exchange. These results were expected since delegating the computation of DH modular exponentiations (in the key agreement mode) leads to more energy savings at the constrained device than offloading signature and encryption operations in the key transport mode. Energy savings can be increased by reducing the duration of listening mode. Using LPL (Low Power Listening) protocols [40], the source node can be temporarily put into a sleep mode when waiting for the protocol to run between proxies and server. These saving can be especially important for the key agreement protocols, where the listening communication cost amounts to more than 50% of the overall energy consumption.

The results also show that the energy costs of the threshold distributed approach in the key agreement mode of IKE are slightly less small than those of the simple distributed approach; contrary to what may have been expected if one had only considered the additional cost of the generation of the polynomial shares. This generation overhead certainly makes the secret distribution more complex, but meanwhile it reduces the energy cost of messages processing at the source node, which receives and deciphers k packets instead of n. These k packets contain shares of DH session key sent from proxies at the end of the protocol exchange to make it possible for the source node to set up the master key.

Concerning the one-pass key transport mode of TLS handshake, the overhead introduced by the addition of redundant parity packets in the threshold distributed approach slightly increases the energy cost of the protocol exchange. On the other hand, the constrained source is not expected to process packets received from proxies so that the introduced overhead is not compensated as in the key agreement mode.

In a nutshell, simulation results prove the viability of the proposed distributed approaches in the studied context of IoT keying, which involves highly resource-constrained nodes such as the TelosB sensor platform. Providing almost equivalent energy costs compared to the simple distributed approach, the threshold distributed approach introduces additional recovery and secrecy properties, both essential for a collaborative protocol.

6. Security analysis

6.1. Key compromising

The collaborative scheme is based on multiple hop-by-hop deliveries of secret fragments, each fragment being sent through a proxy that can therefore access it "in clear". Hence, a first point of focus of the present security analysis consists in dealing with malicious proxies in order to ensure a fair level of security even in case of their presence. The selection of multiple proxy nodes at the constrained device represents the main protection against key compromising, since a single proxy will only get access to a part of the secret – the more numerous the proxies, the smaller the fragment disclosed to each proxy. Selecting the right number of proxies should be a function of the network size and topology, the degree of resilience required against attacks and the quantity of resources that a proxy is devoting to collaborative services. It is evident that choosing a small number of proxies causes a bottleneck and creates performance problems while selecting a high number of proxies increases the communication and, in certain cases, computational overhead during the protocol exchange.

Yet, we assume that proxies may collude by sharing different secret fragments delivered by the constrained node, in order to reassemble them and reconstitute the session key. A first way to counter this potential threat of collusion attack is to base the selection of assisting proxies on a trust model. This model is fed by a system that tracks nodes' behavior in the network and takes into account previous experience, to eventually select well behaving nodes to assist the constrained node and exclude malicious ones from the key establishment process. The assessment of assisting nodes' behavior can be based on direct observations of the proxy or feedbacks collected from the second endpoint of the communication involved in the key establishment

Table 4
Energy costs of cryptographic operations required by the different evaluated approaches on a TelosB processor for for the TLS handshake protocol in key transport mode. (PMK of 48 bytes, AES 128 CBC, HMAC SHA).

	TLS Handshake protocol		IKE protocol	
	Cryptographic operations	Energy cost	Cryptographic operations	Energy cost
Basic approach	verify_CERT + verify_sign + RSA_encrypt_x + RSA_sign_ encrypt_x + compute_Master Key + compute_Finished + verify_Finished	2.1 mJ + 1.2 mJ + 1.6 mJ + 24.43 mJ + 20.92 μJ + 267.1 μJ + 686.58 μJ = **30.30 mJ**	compute_DH$_i$compute_K_{DH} + compute_K_M + compute_sign K_M_encrypt_msg3 + compute_MAC_K_M + verify_MAC_K_M + K_M_decrypt_msg4 + verify_CERT + verify_sign	58.97 mJ + 104.73 mJ + 16.74 μJ + 24.39 mJ + 205.25 μJ + 142.31 μJ + 138.12 μJ + 200.31 μJ + 2.1 mJ + 1.22 mJ = **192.11 mJ**
Distr. approach	verify_CERT + verify_sign + n^*(encrypt_xi + compute_MAC) +compute_Master Key + compute_Finished verify_Finished	2.1 mJ + 1.2 mJ + 5*(2.47 μJ + 16.74 μJ)+20.92 μJ + 267.1 μJ + 573.56 μJ = **4.25 mJ**	n^*(encrypt_a_i + compute_MAC + verify_MAC + decrypt_$g^{b.ai} \bmod p$) + compute_mult_$g^{ai.b}$ + compute_K_M + K_M_encrypt_msg3′ + compute_MAC_K_M + verify_MAC_K_M + K_M_decrypt_msg4′ + n^*(compute_MAC + verify_MAC)	5*(2.47 μJ + 10.46 μJ + 23.02 μJ + 19.78 μJ) + 290 μJ + 16.74 μJ 29.67 μJ 23.02 μJ 18.83 μJ 24.73 μJ 5*(2.1 μJ + 2.1 μJ) = **702.64 μJ**
Threshold distr. approach	verify_CERT + verify_sign + encode_reed_solomon + n^*(encrypt_xi + compute_MAC)+compute_Master Key + compute_Finished verify_Finished	2.1 mJ + 1.2 mJ + 350,6 μJ + 5*(2.47 μJ + 16.74 μJ) + 20.92 μJ + 267.1 μJ + 573.56 μJ = **4.6 mJ**	$n^*(k-1)^*$(comp_mult_$f(i)$) + compute_add_$f(i)$) + n^*(encrypt_$f(i)$ +compute_MAC) + k^*(verify_MAC + decrypt_$g^{b.f(i).ci} \bmod p$) + compute_mult_$g^{b.f(i).ci}$ + compute_K_M + K_M_encrypt_msg3′ + compute_MAC_K_M + verify_MAC_K_M + K_M_decrypt_msg4′ + k^*(compute_MAC + verify_MAC)	5*2*(0.09 μJ + 0.05 μJ) + 5*(2.47 μJ + 10.46 μJ) + 3*(23.02 μJ + 19.78 μJ) + 290 μJ + 16.74 μJ 29.67 μJ 23.02 μJ 18.83 μJ 24.73 μJ 3*(2.1 μJ + 2.1 μJ) **= 610.04 μJ**

Table 5
Power consumption of TelosB at 4 MHz with a transmit power of −5 dB m (from [38]).

	TelosB platform (mW)
Transmit	54
Receive	61
Listen	60

scheme. This latter sends the list of participating nodes that contacted it to assist the key exchange to the constrained node that transmits it in turn to the trust manager. By analyzing the received reports, the trust manager learns about the results of its last assignment decision. It becomes able to detect misbehaving nodes and to refine its selection in the future.

Nevertheless, misidentification of a proxy is always possible and the trust model therefore only mitigates the collusion threat. As a second way to defeat collaboration of malicious nodes during the supporting mechanism, the constrained node may keep a small key fragment of the premaster secret (of a size equivalent to the final session key) that it would transmit later to the server, encrypted with the server public key. For a small fragment (as opposed to the entire premaster secret), the encryption overhead on the constrained node would remain limited. At the same time it would considerably increase the difficulty for an attacker to retrieve the key, unless of course most proxies be malicious. During the key transport mode, if the constrained node relies on 6 proxies, each of which handling a 96-bit fragment of a 512-bit premaster secret (thus one proxy may fail without consequences) and the constrained source node keeps the remaining 32 bits, up to 4 colluding proxies would gain no information on a 128-bit session key generated as the hash of the premaster secret.

Sybil attacks, wherein a single malicious node assumes the identity of multiple nodes, can affect the present solution in that a single proxy may pretend being multiple distinct proxies, in order to retrieve more easily multiple fragments sent by the constrained node. This attack is to be handled by the local trusted entity, which should not establish different secured contexts with different instantiations of the same physical node.

A second point of focus for this security analysis consists in validating the security of each simplex delivery of a key fragment from the constrained node to the server through a proxy Pi. The connection from the constrained node to Pi is secured by a pre-established context and does therefore not suffer from specific vulnerabilities. On the other hand, the connection from Pi to the server is dynamically created and cannot benefit from a pre-existing context between these nodes. Hence, the solution was designed to protect the exchanges between Pi and the server against spoofing and man-in-the-middle attacks: Pi receives the server certificate, and validates its identity mapped to its certified public key. The server identifies Pi through the proof of ownership of the one-time Lamport private key, whose corresponding public key is certified by T (trusted by the server).

6.2. Denial of service

Denial of Service (DoS) attacks against our solution would consist in unfair playing or malicious proxy trying to disrupt the collaborative key establishment protocol by sending no or bogus traffic to the server. Without an adapted protection scheme, a selfish proxy could paralyze the whole system and make the key establishment between the constrained source node and the server fail. This kind of "unfair" play has been carefully considered in the

Table 6
Sent and received bytes in the TLS handshake and IKE protocols.

	TLS handshake protocol (key transport mode)			Internet key exchange protocol		
	Basic approach	Distributed approach	Threshold distributed approach	Basic approach	Integer partition approach	Threshold distributed approach
Sent (bytes)	2367	2095	2095	1568	968	932
Recv (bytes)	4610	3484	3484	1542	1496	1236

Table 7
Communication energy costs on a TelosB processor for the TLS Handshake and IKE protocols.

	TLS Handshake protocol			IKE protocol		
	Basic approach (mJ)	Distributed approach (mJ)	Threshold distributed approach (mJ)	Basic approach (mJ)	Integer partition approach (mJ)	Threshold distributed approach (mJ)
Transmit cost	13.63	12.06	12.06	9.03	5.57	5.36
Receive cost	29.87	22.57	22.57	10	9.69	8
Listen cost	24.06	24.66	24.66	24.24	26.76	26.76
Energy cost	67.56	59.29	59.29	43.27	42.02	40.12

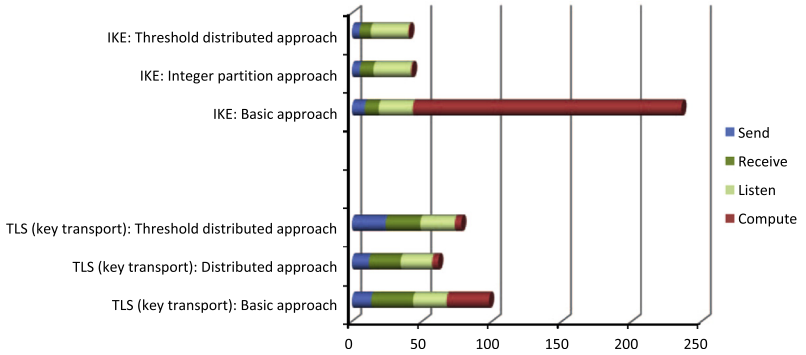


Fig. 11. Overall energy consumption on a TelosB in the two considered key establishment protocols for basic and distributed approaches .

design of our solution. Preventing this type of misbehavior is obviously the keystone of protection against malicious nodes. But, possible recovery from any proxy's misconduct during the process should be the second line of defense. For this reason, we have focused on elaborating both prevention and reaction techniques to overcome this attack. Prevention technique begins at the server. This latter detects the non-cooperative proxies when reassembling the premaster secret or recovering the DH public key and reports a feedback to the constrained source node containing the list of participating proxies. Thereby, the constrained node learns the identities of misbehaving proxies and will prevent their selection in the future.

Recovery is ensured through the use of error correction scheme in the key transport mode and applying the technique of Lagrange polynomial interpolation in the key agreement mode. As explained above, these (k, n) threshold schemes make the establishment of the session key possible even if some malicious proxies refuse to cooperate during the process or in case of unreliable data delivery. In addition to the protection against this non-cooperative behavior, the threshold approach can protect our solution against cheaters that send bogus data. A proxy incorrectly processing a conveyed fragment of the secret key (e.g., replacing the received fragment with a forged one before delivering it to the server) can be identified at the server side. Indeed, this latter can compute different combinations of k messages from the pool of n messages and detect the node providing wrong information Thus, it ensures resiliency of our solution to this type of attack.

Another type of DoS attacks would consist for an attacker to use the mechanism proposed in the present paper to target nodes or systems participating to the solution. Especially, exhaustion attacks could occur if a malicious client node attempts to drain the battery and/or to increase the use of computational resources of proxies, pretending to be needing their assistance. Thus, a trust model of the entities with which they are interacting is required at the proxy side also.

7. Conclusion

This paper presented a novel collaborative approach for key establishment in the context of the IoT, by which a resource-constrained device delegates its expensive computational load to assisting nodes, on a distributed and cooperative basis. In order to enable this collaborative behavior, two distributed techniques have been proposed and carefully designed for both the key transport and key agreement modes. These techniques have then been assessed and compared to basic key exchange standards from the points of view of cryptographic and communication costs. Simulations results first show that our proxy-based scheme significantly increases the energy savings at the constrained device compared to existing standards. They also show that the threshold distribution should be the preferred approach for a constrained node taking part to the collaborative process we propose for key exchange. Next steps in the validation of this proposed work would consist in designing a trust model that manages cooperation between nodes for establishing a community of trusted elements assisting each other. This trust model used to select proxy nodes should be also precisely conceived, in order for it to be as energy-efficient as possible. Here also, heterogeneity of the IoT will be leveraged in order to optimize cognitive operations aimed at producing a global trust model on a distributed basis.

Acknowledgement

This work was financially supported by the EC under Grant agreement FP7-ICT-257521 IoT-A project.

References

[1] F.L. Lewis, Wireless sensor networks, in: D.J. Cook, S.K. Das (Eds.), Smart Environments: Technology, Protocols, and Applications, Wiley, 2004.
[2] W. Geng, S. Talwar, K. Johnsson, N. Himayat, K.D. Johnson, M2M: from mobile to embedded internet, IEEE Commun. Mag. 49 (4) (2011) 36–43.
[3] C. Wietfeld, H. Georg, S. Groening, C. Lewandowski, C. Mueller, J. Schmutzler, Wireless M2M communication networks for smart grid applications, in: Wireless Conference 2011 – Sustainable Wireless Technologies (European Wireless), April 2011.
[4] Y. Zhang, R. Yu, S. Xie, W. Yao, Y. Xiao, M. Guizani, Home M2M networks: architectures, standards, and QoS improvement, IEEE Commun. Mag. 49 (4) (April 2011) 44–52.
[5] A.J. Menezes, S.A. Vanstone, P.C. Van Oorschot, Handbook of Applied Cryptography, CRC Press Inc., Boca Raton, FL, 1996.

[6] W. Diffie, M.E. Hellman, New directions in cryptography, IEEE Trans. Inform. Theory 22 (1976) 644–654.
[7] V. Manral, Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH), IETF RFC 4835, April 2007.
[8] J. Arkko, E. Carrara, F. Lindholm, M. Naslund, K. Norrman, MIKEY: Multimedia Internet KEYing, IETF RFC 3830, August 2004.
[9] P. Eronen, H. Tschofenig, Pre-shared Key Ciphersuites for Transport Layer Security (TLS), IETF RFC 4279, December 2005.
[10] T. Dierks, E. Rescorla, The Transport Layer Security (TLS) Protocol Version 1.2, IETF RFC 5246, August 2008.
[11] C. Kaufman, Internet Key Exchange (IKEv2) Protocol, IETF RFC 4306, December 2005.
[12] R. Moskowitz, Host Identity Protocol Architecture, draft-ietf-hip-rfc4423-bis-03 (IETF Work in Progress), September 2011.
[13] V. Cakulev, G. Sundarm, I. Broustis, IBAKE: Identity-based Authenticated Key Exchange, IETF RFC 6539, March 2012.
[14] H. Chan, A. Perrig, D. Song, Key distribution techniques for sensor networks, in: T. Znati et al. (Eds.), Wireless Sensor Networks, Springer, 2004, pp. 277–303 (Chapter 13).
[15] D. Liu, P. Ning, Location-based pairwise key establishments for static sensor networks, in: Proceedings of the 1st ACM Workshop on Security of Ad Hoc and Sensor Networks (CCS '03), October 2003, pp. 72–82.
[16] W. Du, J. Deng, Y.S. Han, S. Chen, P.K. Varshney, A key management scheme for wireless sensor networks using deployment knowledge, in: Proceedings of IEEE INFOCOM'04. 2004.
[17] S. Schmidt, H. Krahn, S. Fischer, D. Watjen, A security architecture for mobile wireless sensor networks, in: Proceedings of the 1st European Workshop on Security in Ad-Hoc and Sensor Networks (ESAS'04) (Heidelberg, Germany), vol. 3313, August 2004.
[18] N.R. Potlapally, S. Ravi, A. Raghunathan, N.K. Jha, A study of the energy consumption characteristics of cryptographic algorithms and security protocols, IEEE Trans. Mobile Comput. (2006) 128–143.
[19] V. Gupta, M. Millard, S. Fung, Yu Zhu, N. Gura, H. Eberle, S. Chang Shantz, Sizzle: a standards-based end-to-end security architecture for the embedded internet, in: Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications, March 08–12, 2005, pp. 247–256.
[20] ANSI X9.62, Elliptic Curve Key Agreement and Key Transport Protocols, American Bankers Association, 1999.
[21] ANSI X9.63, The Elliptic Curve Digital Signature Algorithm, American Bankers Association, 1999.
[22] W. Jung et al., SSL-based lightweight security of IP-based wireless sensor networks, in: International Conference on Advanced Information Networking and Applications Workshop, 2009.
[23] V. Nagalakshmi, I. Rameshbabu, P.S. Avadhani, Modified protocols for internet key exchange (IKE) using public encryption key and signature keys, in: Proc. of the Eighth International Conference on Information Technology: New Generations, 2011, pp. 376–381.
[24] R. Sangram, G.P. Biswas, Establishment of ECC-based initial secrecy usable for IKE implementation, in: Lecture Notes in Engineering and Computer, Science, 2012, pp. 530–535.
[25] A. Liu, P. Ning, TinyECC: A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks, Technical Report TR-2007-36, North Carolina State University, Department of Computer, Science, 2007.
[26] L. Atzori, A. Iera, G. Morabito, The Internet of Things: A Survey, Elsevier Computer Networks, 2010.
[27] R. Merkle, Secrecy, Authentication, and Public Key Systems, Ph.D. Dissertation, Dept. of Electrical Engineering, Stanford Univ., 1979.
[28] N. Fazio, A. Nicolosi, Cryptographic accumulators: definitions, constructions and applications, in: Technical Report, 2002.
[29] L. Lamport, Constructing digital signatures from one-way function, in: Technical Report SRI-CLS-98, SRI international, October 1979.
[30] S. Seys, B. Preneel, Power consumption evaluation of efficient digital signature schemes for low power devices, in: Proceedings of the 2005 IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (IEEE WiMob 2005), 2005, pp. 79–86.
[31] M. Watson, Basic Forward Error Correction (FEC) Scheme, RFC 5445, March 2009.
[32] A. Brusilovsky, I. Faynberg, Z. Zeltsan, Password-Authenticated Key (PAK) Diffie–Hellman Exchange, RFC 5683, February 2010.
[33] A. Shamir, How to share a secret, Commun. ACM 22 (1979) 612–613. November.
[34] W. Dai, Crypto++ Library 5.6.0. <http://www.cryptopp.com>.
[35] J. Lacan, V. Roca, J. Peltotalo, Reed–Solomon Forward Error Correction (FEC) Schemes, IETF RFC 5510, April 2009.
[36] IT++ Library. <http://itpp.sourceforge.net/current/>.
[37] A. Wander, N. Gura, H. Eberle, V. Gupta, S.C. Shantz, Energy analysis of public-key cryptography for wireless sensor networks, in: Third IEEE International Conference on Pervasive Computing and Communications, 2005, pp. 324–328.
[38] G. De Meulenaer, F. Gosset, F.-X. Standaert, O. Pereira, On the energy cost of communication and cryptography in wireless sensor networks, in: Proceedings of IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WIMOB 2008), 2008.
[39] J. Paek, K. Chintalapudi, R. Govindan, J. Caffrey, S. Masri, A Wireless Sensor Network for Structural Health Monitoring: Performance and Experience, IEEE Computer Society, Washington DC, USA, 2005.
[40] C. Merlin, W. Heinzelman, Duty cycle control for low-power-listening mac protocols, in: MASS, 2008.

Yosra Ben Saied, CEA-LIST Yosra Ben Saied graduated in 2010 from the Higher School of Communications of Tunis (Sup'Com), where she obtained her National Diploma in Telecommunications Engineering and Master's degree in Telecommunications. In 2010 she joined as a PhD student the Laboratory of Communicating Systems (LSC) at French Atomic Energy Commission. Her research activities consist in developing network security solutions for constrained systems. She especially focuses on cognitive and collaborative mechanisms.

Alexis Olivereau, CEA-LIST Alexis OLIVEREAU graduated from École Nationale Supérieure de l'Électronique et ses Applications, Cergy, France in 2000. Between 2000 and 2008 he has been a research engineer in the Motorola Labs research center of Paris, France where he worked on networking security, developing novel protocols for IP-based architectures in the framework of mobile Internet. He participated in various European research projects and earned Motorola "Gold Badge" distinction for his patents filing. He joined the Laboratory of Communicating Systems (LSC) of CEA-LIST in January 2009 as a researcher and is now working on security and privacy aspects of communications in machine-to-machine and cloud environments.

Djamal Zeglache, Telecom SudParis Djamal Zeghlache Professor graduated from SMU in Dallas, Texas in 1987 with a Ph.D. in Electrical Engineering and joined the same year Cleveland State University as an Assistant Professor. In 1990 and 1991 he worked with the NASA Lewis Research Centre on mobile satellite terminals, systems and applications. In 1992 he joined the Networks and Services Department at Telecom SudParis of Institut Telecom where he currently acts as Professor and Head of the Wireless Networks and Multimedia Services Department. Professor Zeghlache is also acting Dean of Research of Telecom SudParis. He co-authored around one hundred publications in ranked international conferences and journals and was an editor for IEEE Transactions on Wireless. His interests and research activities span a broad spectrum related to fixed and wireless networks and services. The current focus is on network architectures, protocols and interfaces to ensure smooth evolution towards loosely coupled future Internet, cloud networking and cloud architectures. He is currently addressing inter-domain cooperation and federation challenges for these networks, related modeling for resource optimization of infrastructures and platforms offered as a service to users and providers.

Maryline Laurent, Telecom SudParis Maryline Laurent, PhD works as a professor at Telecom SudParis, Mines-Telecom Institute, and is the head of the research team R3S (Network, Systems, Services, Security) of the French CNRS UMR 5157 SAMOVAR. Her main topics of interest are related to network security and privacy, including IPv6, mesh/ad hoc networks, RFID systems, social networks and identity management. She chaired the Security track of the IFIP International Conference on New Technologies, Mobility and Security NTMS 2011, and ICSNA 2011 (International Conference on Secure Networking and Applications). She is currently coediting a book on identity management, Ed. Lavoisier, and she is editor of the special issue on ‘‘Privacy-aware electronic society’’, Annals of Telecommunications.