HAL
open science

OSCAR: Object Security Architecture for the Internet of Things

Malisa Vucinic, Bernard Tourancheau, Franck Rousseau, Andrzej Duda, Laurent Damon, Roberto Guizzetti

To cite this version:

Malisa Vucinic, Bernard Tourancheau, Franck Rousseau, Andrzej Duda, Laurent Damon, et al.. OSCAR: Object Security Architecture for the Internet of Things. A World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2014 IEEE 15th International Symposium on, Jun 2014, Sydney, Australia. 10.1109/WoWMoM.2014.6918975 . hal-00985976v2

HAL Id: hal-00985976

https://inria.hal.science/hal-00985976v2

Submitted on 18 Dec 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

OSCAR: Object Security Architecture for the Internet of Things

Mališa Vučinić*¶, Bernard Tourancheau*, Franck Rousseau*, Andrzej Duda*, Laurent Damon¶, Roberto Guizzetti¶
*Grenoble Alps University, CNRS Grenoble Informatics Laboratory UMR 5217, France
¶STMicroelectronics, Crolles, France
Email: {firstname.lastname}@imag.fr, {firstname.lastname}@st.com

***Abstract*—Billions of smart, but constrained objects wirelessly connected to the global network require novel paradigms in network design. New protocol standards, tailored to constrained devices, have been designed taking into account requirements such as asynchronous application traffic, need for caching, and group communication. The existing connection-oriented security architecture is not able to keep up—first, in terms of the supported features, but also in terms of the scale and resulting latency on small constrained devices. In this paper, we propose an architecture that leverages the security concepts both from content-centric and traditional connection-oriented approaches. We rely on secure channels established by means of (D)TLS for key exchange, but we get rid of the notion of the "state" among communicating entities. We provide a mechanism to protect from replay attacks by coupling our scheme with the CoAP application protocol. Our object-based security architecture (OSCAR) intrinsically supports caching and multicast, and does not affect the radio duty-cycling operation of constrained objects. We evaluate OSCAR in two cases: 802.15.4 Low Power and Lossy Networks (LLN) and Machine-to-Machine (M2M) communication for two different hardware platforms and MAC layers on a real testbed and using the Cooja emulator. We show significant energy savings at constrained servers and reasonable delays. We also discuss the applicability of OSCAR to Smart City deployments.**

I. Introduction

The long awaited Internet of Things (IoT) has never been closer. The upper layers of the IP protocol stack for constrained devices are being fine-shaped and the gaps between IETF and IEEE standards are being bridged. Security remains a concern. Well designed IP security protocol suites have been ported to constrained devices of IoT. Their core design assumptions, however, build upon the connection-oriented trust model that poorly fits the IoT requirements. Nevertheless, past experiences have shown that designing a security protocol right is a tough and error-prone process.

Research efforts towards the secure IoT have thus mostly concerned designing lightweight variants and porting them to constrained devices [1], [2], [3], [4], which led to a situation where the nearly standardized Constrained Application Protocol (CoAP) [5] fully supports the application requirements, but security does not keep up. Smart devices, due to their severe energy and memory constraints, heavily rely on group communication, asynchronous traffic, and caching. Supporting a variety of existing security protocols/mechanisms to specifically target these requirements is practically impossible due to memory limitations. IETF has thus taken a position [5] to reuse Datagram Transport Layer Security (DTLS), the all-round point-to-point security protocol, to secure the communication channel between a constrained node running a CoAP server and a client.

Apart from the inherent incompatibility with multicast traffic and caching, the plain DTLS approach has an important impact on scalability. Namely, memory limitations of constrained servers restrict the total number of handled security sessions. In IoT scenarios, such as smart city deployments, where a large number of clients per constrained server is expected, the limitations lead to a considerable load on the server to handle security associations with each client. The load translates into increased energy consumption and a shortened lifetime of devices.

We address this problem from a networking perspective and follow the Representational State Transfer (REST) architecture model [6] to remove the notion of the state between a server and a client even in terms of security. We achieve this by leveraging the concept of object security that protects the information content itself. We couple the object security principles with the capability-based access control to provide communication confidentiality and protect from replay attacks. Yet, we fully leverage a vast amount of work behind the (D)TLS protocol and use secure channels for authenticated key distribution.

The main contributions of the paper are the following:

- a new scalable security architecture for IoT that jointly provides end-to-end security (E2E) and access control, decouples confidentiality and authenticity trust domains, and intrinsically supports multicast, asynchronous traffic, and caching,
- an evaluation of the architecture in a constrained Machine-to-Machine (M2M) scenario for two hardware platforms and MAC layers, on a real testbed and in the instruction level emulator of Cooja, demonstrating performance benefits with an increasing number of clients.

The paper is organized as follows. We discuss the current Internet trust model and the requirements of IoT applications in Section II. We provide a detailed description of the proposed architecture in Section III, discuss security considerations in Section IV, and evaluate it in Section V. Section VI summarizes the related work. We conclude and discuss the future work in Section VII.

II. Internet Trust Model and the IoT Requirements

The fact that the Internet had been designed to facilitate host to host communication has had direct repercussions on the security design. Namely, security followed the model by placing the trust on end points and securing the communication channel. As applications evolved, the Internet has become a content distribution network leveraging the legacy client-server architecture. This paradigm has led to substantial research efforts on future Internet architectures [7], [8]. Our work leverages their contributions and applies the general concepts with the goal to provide a robust, but flexible security approach to IoT and its traffic requirements.

As discussed by Smetters and Jacobson [9], the host oriented paradigm has a direct consequence on trust—its transitivity: once a logical connection between the hosts is closed, the trust in the information is gone. The model serves very well typical e-commerce, e-banking, or IP telephony applications, because the trust in the information is implicitly dependent on the trust of the communicating entities *during the connection time*.

The difficulty arises once the notion of a connection disappears. As stressed by Modadugu and Rescorla [10], DNS is purposely secured with the application level extension DNSSEC and not with a connection-oriented protocol, such as DTLS. Electronic mail, passing multiple application level gateways and without clear connection between end points, is secured with S/MIME or PGP. Applications encompassing IoT emerge as another example, because:

- *Application traffic is asynchronous.* Servers (event detectors, monitoring sensors, smart meters) notify their clients (subscribers) of physical state changes as they happen. Clients send commands to actuating devices asynchronously as the changes in the environment are observed. DNS traffic is a good parallel as it is triggered by asynchronous human actions.
- *Caching is a must.* Severe energy constraints lead to servers being asleep more than 99% of the time. As an already supported (without security) and intuitive mechanism, caching at untrusted intermediaries is a way to keep applications running independently. A similar problem is faced with electronic mails, as they are stored at untrusted servers until delivery.
- *Group communication is frequent.* Commonly, clients instruct a subset of all devices to perform an action, for example to turn off all lights on n^{th} floor or to update the firmware. To achieve this, IPv6 multicast and UDP are exploited bearing no connection state between end points.

Typical Web applications are built around a single logical server and multiple clients [6]. As a consequence, access control is often done within the server side application, once the client has been authenticated. IoT reverses this paradigm by having many devices serving as servers and possibly many clients, taking part in the same application. More importantly, servers are significantly resource constrained, which results in the minimization of the server side functionality. Subsequently, access control becomes a distributed problem, especially when taking into account the recent efforts of decoupling the sensor network infrastructure from applications [11], [12]. Furthermore, applications have emerged that use local databases to store parts of collected data [13].

Recognizing these requirements, it is clear that the connection-oriented trust model is not the best fit for the actual needs of IoT. It is true that with different sorts of connection time tweaking and keep-alive messages we could squeeze in connection-oriented security protocols and work around the asynchronous traffic requirement. Aside the overhead, this would still provide us only with the communication channel security. To support caching, we would need to trust the intermediate nodes/proxies to store the data. Note that we deal with devices physically accessible to anyone. To support group communications, we would need to open separate secure connections among group members and/or add additional protocols on top of them, which effectively provides redundant security services necessary for use cases. Such a solution is not a long term approach.

Nevertheless, we do not argue that we should ditch well studied connection-oriented security protocols from the IoT picture. In fact, OSCAR relies on secure and authenticated channels established by means of DTLS for key distribution: our approach brings together the concepts of connection-oriented security with those of content-centric networking [7].

III. OSCAR

We argue that we can meet the discussed requirements by leveraging the benefits of the "object security" concept. At the same time, we can provide much greater flexibility to the system as a whole.

A. Technological Trends and Design Goals

Future trends are hard to predict, but a decade of research on Wireless Sensor Networks has given us a lot of insights. Accordingly, we draw the following conclusions that guide our design:

- Constraints on energy are almost constant. Without a breakthrough in chemical engineering, the available energy is expected to remain the main constraint for IoT devices.
- Available memory for embedded devices slowly increases. However, due to the economical and energy cost caused by leakage in SoC, we expect that memory will remain limited and a determining factor for the unit price.
- Processing capabilities constantly increase even for ultra low power micro controllers. Thus, we do not see the processing power as a limiting constraint in the future.

Apart of sleep mode leakages, the energy consumption is mainly caused by radio communications. Thus, our primary design goal is to minimize the number of extra frames/packets that need to be transmitted or received for pure security purposes. We achieve this goal by leveraging the benefits of

public key cryptography, sparse traffic patterns within local constrained networks, and messages of a limited size—we trade the radio usage for a higher computation load.

B. Producer-Consumer Model

We can abstract IoT, its sensors and actuators, as an interface to the physical world. Decision takers (human users, intelligence centers, or constrained actuating devices themselves) base their reasoning on input data coming from the sensed physical phenomena. The relation between enforced decisions and sensed phenomena is *many to many*—a single measurement often affects multiple decisions and a single decision may be based on many different phenomena. Consider for instance a traffic control application in a Smart City. A traffic light management subsystem may use the current traffic intensity and pollution readings from all over the city as input data for control decisions. At the same time, local readings may influence decisions made on luminosity of nearby street lights.

We believe that the producer-consumer model represents well our problem also in terms of security. Producers (smart meters, traditional sensors, motion detectors, etc.) feed consumers with the required information. Consumers (actuating devices, collection centers, human users) gather up the information and may further *generate* actions. Actually, the inspiration for the use of the model comes from Cloud Computing and a recent work by Jung *et al.* on data access control [14]. An important difference, however, is that producers in the IoT case are not access control decision makers for the content they generate, which is rather a policy of the network operator.

Producers should, thus, care about generating and securing the content or in the REST terminology, the resource representations, and not about consumers. Consumers with appropriate access privileges should make sure they can make use of the fetched content by decrypting and authenticating its validity.

Following the discussion, we believe that the extent of security tasks performed by producers should be minimized—producers should not waste precious resources on exchanging security handshake messages with each consumer. From the producer perspective, there are two main reasons for this situation:

- *Resource representations are minimal in size.* The generated content, *i.e.*, the resource representations are typically the measurements of physical quantities or different states of a device with possibly additional information such as location and a timestamp, which very often makes them smaller in size than individual messages exchanged during a security handshake. As a consequence, responding with an access-protected resource representation is cheaper than performing multiple RTT handshakes.
- *Due to the physical constraints, the number of supported cryptographic ciphers is limited.* Indeed, constrained devices often have a single supported cipher suite (selected at the compile time). This fact reverses the paradigm encountered in the Internet where one of the security concerns during the handshake is the downgrade attack (the attacker forces two parties to use the weakest common cipher by altering the set supported by the client). The motivation behind the attack is the assumption that the client cipher set is just a subset of those supported by a resource rich server. With the reversed paradigm, as in the IoT case, the motivation for the attack fades away.

One of our goals is to offload the burden of authentication from constrained servers and to place it on more powerful devices. Such semi-trusted third parties could be physically secured nodes in the network and/or hosts in the Cloud. Their role would be to authenticate individual consumers and share with them appropriate *access secrets* and necessary certificates. We define an "access secret" as an access token from which symmetric encryption keys are derived. Later, consumers can fetch the protected content either from intermediate proxies or directly from producers.

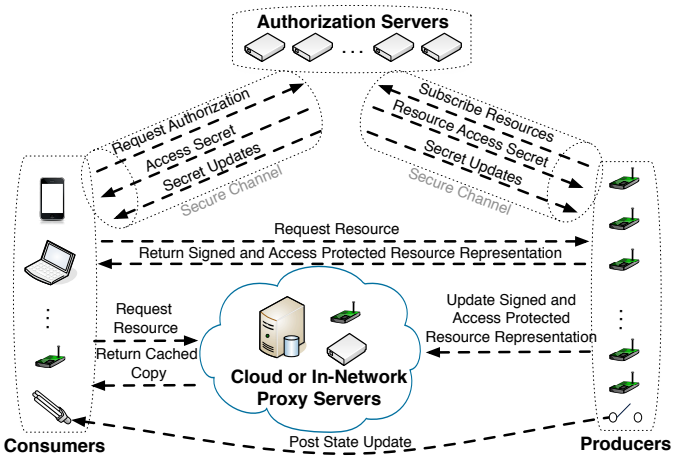


Fig. 1. OSCAR, a producer-consumer security model for IoT based on object security.

We thus consider separately *in terms of trust* two commonly interleaved security services:

- *Confidentiality is used as a means to provide capability-based access control and a protection against eavesdropping.* As a consequence, third parties in charge of authorization need to be trusted.
- *Authenticity and integrity of content is tied to the host.* Consumers independently decide if they will trust the source *for the provided content*. For example, temperature readings would need to be signed by a server that is certified to have a temperature sensor and to be deployed in the wanted physical location.

This approach could be interpreted as we disclose true E2E confidentiality to third parties. However, even in the classical TLS scenario, the authorization authority running on a server has potential access to all the information flowing on the secured channel. Note that authenticity and integrity of information are not affected and the traditional E2E properties are preserved.

Fig. 1 illustrates the abstract model and the logical interactions. Notice that from the perspective of a producer, a direct request-response interaction with a consumer is handled

equally as in the case of asynchronous updates. More precisely, producers locally keep cached and secured resource representations, *i.e.* signed objects, and use them to feed different types of consumers. In essence, we remove any notion of a logical association between a producer and a consumer.

In the following, we assume that producers and consumers already possess necessary certificates. During the resource subscription phase, producers publish their certificates to Authorization Servers. If a certain certificate is required for signature verification purposes, consumers can fetch it from them. In this way, we remove the burden of certificate transmission from constrained servers to their multiple clients.

C. Fitting the Concept with the REST Architecture and CoAP

While object security is traditionally used by applications themselves, we discuss here the coupling with CoAP, the RESTful application protocol. By doing so, we aim to make a bound between secured objects and the underlying communication protocol, in order to protect against network adversaries in a stateless manner. Note that in the rest of the paper, we use the REST terminology and refer to producers as REST servers and consumers as REST clients. In IoT deployments, however, the same physical device often plays both roles.

We abstract the access secrets as REST resources, which allows using the idempotent PUT method to create or update them. Servers, then, allow the change only if the enclosed object, *i.e.* a new access secret, has been signed by the trusted authority. The relation between j^{th} access secret S_j and i^{th} resource R_i is dependent on authorization policies and the desired level of confidentiality. It is a part of resource R_i itself allowing for different confidentiality and access right resource groups (cf. Figure 2).

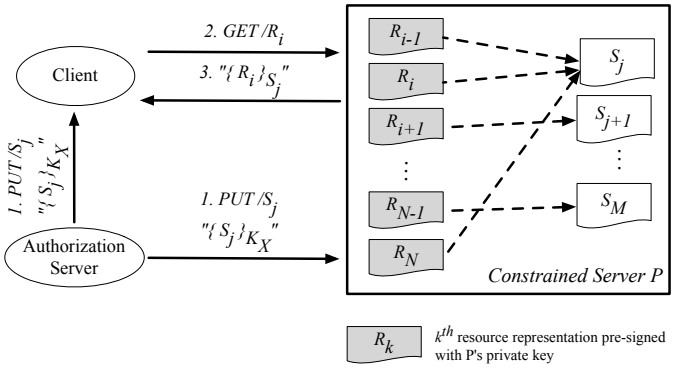


Fig. 2. Coupling OSCAR with the RESTful architecture of CoAP. Note that server-side digital signing operations are performed offline. $\{X\}_K$ denotes the online symmetric encryption of X with the key derived from K. Encryption key K_X of a new access secret may depend on the key management scheme.

In the following, we discuss two important aspects from the communication point of view.

1) Replay protection: Protecting against replay attacks requires a state between end-points, which contradicts our goal to provide a stateless approach to application security. However, we exploit the fact that CoAP has been designed to run over UDP and so, detect duplicates using a 16-bit

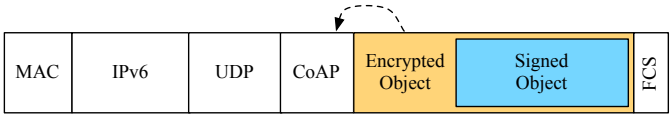


Fig. 3. Secured objects within the IoT network stack. The arrow represents the binding of the object encryption key with the underlying CoAP header. In this configuration, signed object is regarded as the payload of the encrypted object, and is therefore encrypted.

MessageID in the protocol header. We use this variable as salt to derive the content encryption key, when responding to the current request. Furthermore, in order to avoid derivation of an identical content encryption key among different senders in a possible group, we bind the key to the identity of the sender, as discussed by Keoh *et al.* [15]. We use the unique identifier in the sender certificate known to all communicating parties.

More precisely, symmetric content encryption key k_j is derived from access secret S_j, current MessageID, and sender's unique identifier as:

$$k_j = f(S_j, MessageID, senderID), \tag{1}$$

where $f()$ is a generic pseudo-random function. Note that we keep the authenticity of the content intact and use symmetric encryption as a means to fight the replay and provide access control. We illustrate this in Fig. 3. The replayed content will be detected as the derived decryption key will be different from the original key.

However, this approach is vulnerable to replay attacks with substantially delayed content, *i.e.* once the CoAP client/server looses the MessageID context with communicating parties. To fight this, we rely on updates of the access secret, provided by the key management scheme.

2) Cipher negotiation: As discussed in Section III-B, there is no motivation in traditional downgrade attacks. The problem of cipher negotiation then becomes trivial and is equivalent to content type negotiation in a client-server interaction. Recall that a client in a REST like application transfer protocol such as HTTP or CoAP, expresses its interest in content with a *GET* request. The request contains an optional "Accept" header carrying preferred content types. If capable, server responds with the content supported by the client. Therefore, to solve the interoperability issues, we require an additional accept option carrying supported ciphers.

D. Cryptographic Overhead

OSCAR ensures authenticity and integrity of the content by leveraging digital signatures, which may seem surprising as we target constrained devices with limited CPU resources. However, the use of public key operations at the level of the content allows us to decouple the server-side cryptographic overhead from network communication: constrained servers are able to update their resource representations whenever it suits their schedule (take for example energy harvested devices) and more importantly, while the radio transceiver is turned off. The burden of digital signature verification is then

put on clients, as they should not consume the information before verifying its authenticity.

The approach may seem surprising, but the evaluation results in Section V suggest that Elliptic curve cryptography (ECC) public key operations are actually less expensive than performing the pre-shared key DTLS handshake with every client, even in M2M scenarios, where both servers and clients are constrained.

Confidentiality of the content is ensured with symmetric encryption performed on a per-response basis by servers. For replay protection, OSCAR requires an additional per-response key derivation with typically lightweight cryptographic pseudo-random functions.

IV. Security Considerations

Denial of Service: OSCAR takes a non-traditional approach to fight Denial of Service. It builds upon the assumption that typical IoT resource representations are small in size (individual measurements of physical quantities, actuator state changes) and directly responds to requests with access-protected resource representations. Moreover, it does not keep any state between communicating entities, which we find particularly important to fight memory exhaustion attacks. Note also that since server-side digital signing operations are done offline, the intensity of incoming traffic is not correlated with asymmetric cryptographic overhead.

Confidentiality: As content encryption keys are derived from access secrets, OSCAR provides confidentiality within the resource access right group. Actual security properties are dependent on the encryption algorithm used. Note that an adversary able to compromise the Authorization Servers, may only obtain eavesdropping capabilities—E2E integrity and authenticity properties are preserved.

If the mutual trust among clients in terms of confidentiality is not desired, OSCAR puts the burden on the key management scheme running on Authorization Servers. One such example would be the use of a recently proposed batch-based group key management protocol [16], where clients would be given cryptographic material corresponding to descendants in the binary tree of the actual access secret on a server. However, this would require additional signaling of the supported access secret in the GET request.

Replay Protection: OSCAR protects from replay at the level of the content by using an encryption key that is a function of the MessageID from the underlying CoAP header. The detection of replay attacks performed at lower network layers depends on the CoAP duplicate detection mechanism. However, it is important to stress that the current CoAP draft, as is, would not provide robust protection in security terms. Therefore, successful coupling of OSCAR with CoAP would require additional clarifications and specifications to the duplicate detection mechanism.

Another concern with respect to the replay attack is a malicious adversary within the resource access right group in case of asynchronous traffic. Such an adversary is able to asynchronously inject old resource representations making other members of the group believe they are fresh (if within the content itself, there is no means allowing the detection of an old reading/command, *i.e.* a timestamp). Protection against such adversary would require the use of a key management scheme that would provide different access secret cryptographic material on the constrained server and individual clients, as discussed above.

V. Performance Evaluation

We have implemented an object security software library tailored for constrained devices and the Contiki operating system that builds upon the open source implementation of ECC cryptographic primitives—TinyECC (ContikiECC). We use AES-CCM* as the symmetric encryption algorithm. The library supports creation, parsing, and verification of "encrypted" and "signed" object types. A certificate is then just a particular type of a "signed object" with a pre-defined format. Objects can be nested within each other to support the use case illustrated in Fig. 3. We have coupled the object security library with Erbium CoAP, a default CoAP implementation for Contiki (version 07) to add cipher suite negotiation capabilities (cf. Section III-C).

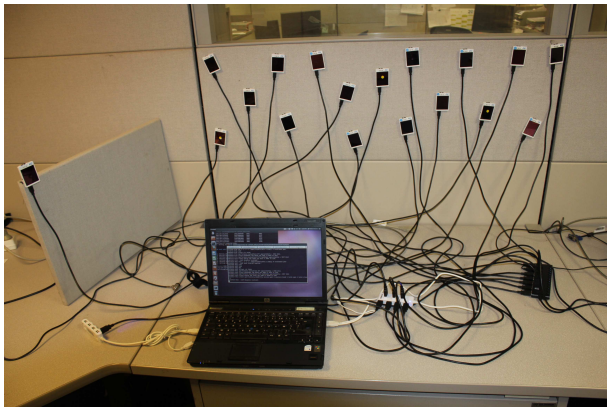


Fig. 4. Testbench with 18 energy-harvested ST GreenNet nodes in Crolles, France. 16 nodes are CoAP clients and one node is the CoAP server. The node on the far left is the PAN coordinator in the 802.15.4 beacon-enabled network. Nodes are connected to USB for the collection of experiment traces.

Note that potential nesting of signed objects enables many additional features that may be very useful for IoT use cases. For instance, a network gateway could add a global timestamp or location information to a signed object coming from a constrained node (constrained devices often do not have this information locally).

We evaluate two important aspects of OSCAR: 1) Elliptic Curve Digital Signature Algorithm (ECDSA) computation overhead on constrained devices, 2) scalability in M2M communication scenarios. Evaluations are performed for two hardware platforms that catch peculiarities of two generations of IoT devices:

- WiSMote platform based on 16-bit MSP430 (series 5) MCU with 16 KB of RAM and 802.15.4-compatible CC2520 radio transceiver. WiSMote related results are

obtained using the instruction level MSP430 emulator MSPSim and the Contiki simulator Cooja, as we did not have enough real platforms needed for our experiments. However, we have confronted emulated measurements of ECDSA overhead in Cooja with those obtained on real WiSMote hardware and we have measured a maximum error of 2.67%.

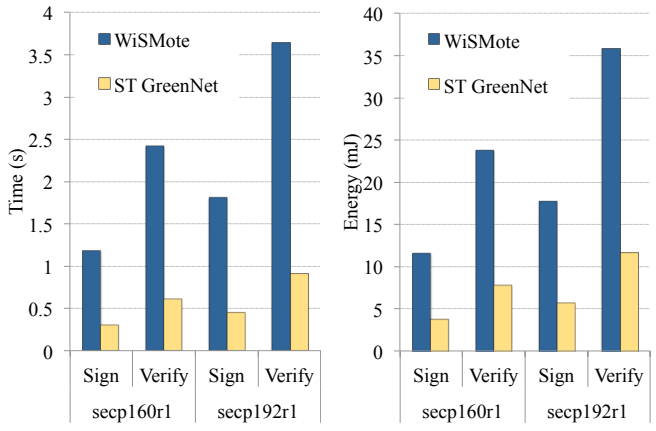
- ST GreenNet tag, an energy-harvested prototype platform from STMicroelectronics (ST) based on an ultra low power 32-bit ARM Cortex-M3 MCU (STM32L) with 32 KB of RAM and an 802.15.4 radio transceiver. ST GreenNet results are obtained from real hardware.

To eliminate the effect of a variable CPU frequency on results, we have configured both platforms at 21.3 MHz. MSP430 series 5 may be configured up to 24 MHz, while the STM32L can go up to 32 MHz. Both computation time (inversely proportional) and CPU energy consumption (directly proportional) are linearly dependent on frequency.

We estimate energy consumption using Energest, a Contiki per-component profiling tool. Energest effectively measures the time spent by different components on a platform in a given state (for instance, the time CPU spent in active or low power mode; radio transceiver in RX or TX). These values are converted to energy by multiplying with the constant operating voltage (we used 2.8 V) and the current draw values from appropriate data sheets.

A. ECDSA Computation Overhead

Figs. 5(a) and 5(b) present computation and energy benchmarks of the ECDSA primitives (secp160r1 and secp192r1 elliptic curves) on WiSMote and ST GreenNet platforms. We can see that the use of a 32-bit MCU reduces the computation time by a factor of 4, which translates into a reduction in the consumed energy by a factor of 3.084 (as the 32-bit STM32L consumes 29.7% more than 16-bit MSP430 in active mode).



(a) Computation time. (b) Energy consumption at 2.8V.

Fig. 5. ECDSA computation and energy benchmarks at 21.3 MHz for 16-bit (WiSMote) and 32-bit (ST GreenNet) hardware platforms. We use TinyECC (ContikiECC) open source library. Message size is of 25 bytes.

Figs. 5(a) and 5(b) strongly support our initial design assumption on processing capabilities (cf. Section III-A). Whatsoever, we expect that further advancements in nanotechnology will additionally reduce the energy computation cost for low power MCUs.

Still, computation overheads ranging from 0.3 to 0.9 seconds for the 32-bit platform and from 1.18 to 3.63 seconds for the 16-bit platform, at the first sight seem like a huge price to pay. In fact, Hummen *et al.* argue that for this reason, the number of public key operations should be minimized during the security handshake [17]. OSCAR, however, compensates for this overhead by removing the radio energy cost of the security handshake with every client.

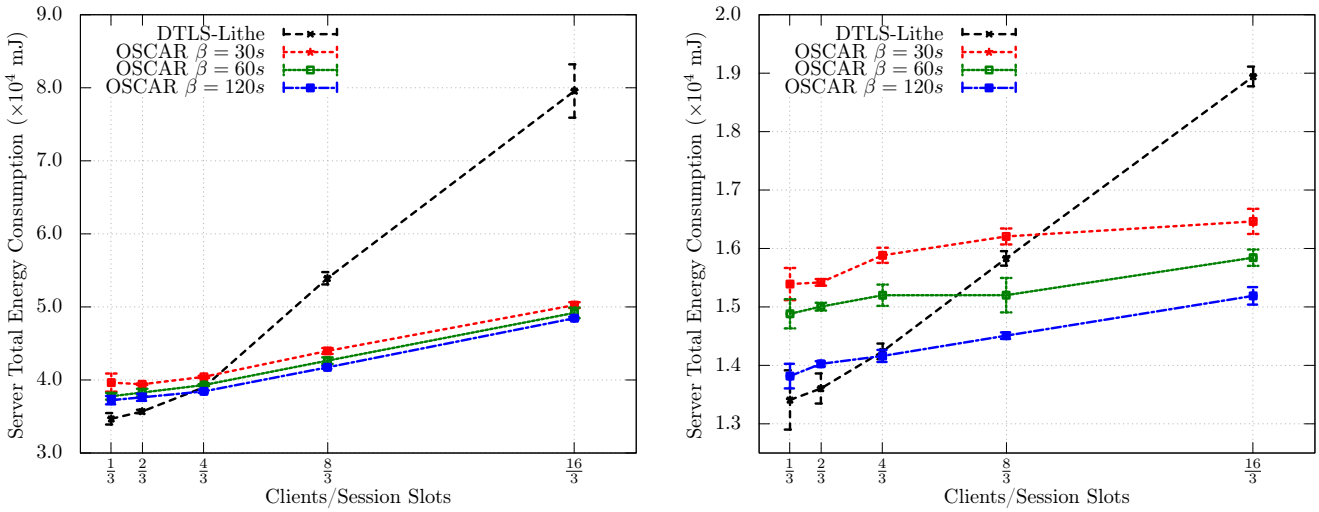
In the following section, our goal is to determine if OSCAR and the heavy use of ECC public key primitives outperform a connection-oriented approach with DTLS that uses only lightweight symmetric key operations during the handshake.

B. Scalability

We study scalability as a function of the ratio between the total number of clients and a maximum number of open DTLS sessions at a constrained server (due to memory limitations, constrained servers have a limited number of DTLS session "slots"). We have followed the guideline on practical issues with DTLS (Section 2.1 [18]) and extended the TinyDTLS implementation with the Least Recently Used (LRU) session closure algorithm. The server immediately releases memory and sends a closing alert to the LRU session as soon as a new client has demonstrated good intentions by retransmitting the stateless cookie in the ClientHello message (recall the DTLS handshake). Therefore, the handshake with the new client proceeds immediately. Clients keep their sessions open as long as possible, *i.e.* until they receive the closing alert from the server.

The maximum number of DTLS session slots is dependent on platform memory capabilities and actual application memory requirements. With the full IPv6 networking stack of Contiki and a simple application for evaluation purposes, we were able to have a maximum of 3 session slots on WiSMote (TinyDTLS implementation). However, as stressed out, this number should not be generalized as it depends on the implementation specifics of an application and the operating system. We have used the same number of slots on the ST GreenNet platform as well to have comparable results. Although a higher number of slots would be available due to a larger memory, the results in terms of the ratio do not loose generality.

Table I shows the configuration of the two platforms. Note that we use two different Radio Duty Cycling (RDC) protocols. Thus, we demonstrate the performance of OSCAR and DTLS running on top of asynchronous (X-MAC) and synchronous (beacon-enabled IEEE 802.15.4) RDC protocols, which covers the vast majority of IoT use cases. We set the Beacon Interval of beacon-enabled 802.15.4 to 122.88 ms to have comparable delays with X-MAC (default channel check rate of 8 Hz). Simulations in Cooja were performed for a star



(a) WiSMote

(b) ST GreenNet

Fig. 6. Constrained server total energy consumption over 3 hours.

network topology with the CoAP server being the central node (radio neighbor for each client and preferred parent in the RPL DODAG). Note that due to the specifics of beacon-enabled 802.15.4, one node in the network has a mere role of being the PAN coordinator and transmitting periodic beacons. Other nodes in the network associate with it (L2 operation), which effectively introduces an extra hop between CoAP clients and the CoAP server, in respect to the network evaluated in Cooja (cf. Fig. 4).

In both scenarios, nodes run a typical IPv6 networking stack over 802.15.4 (CoAP, UDP, IPv6, 6LoWPAN). We use a recent 6LoWPAN compression scheme of DTLS named Lithe [2] to maximize its performance. In the case of ST GreenNet platform, we use the proprietary implementation of beacon-enabled 802.15.4, as in our previous work [19], [20]. CoAP clients send a single GET request for a resource on the server according to the exponential distribution with a mean of 0.5 requests per minute. If the DTLS session is found open, the request is sent directly without waiting for the handshake to complete. If not, the client first performs a DTLS handshake with the server. Responses contain an abstract resource representation with 25 byte length. In case of OSCAR, this representation is transferred as the appropriate encrypted and signed object type.

An important aspect for performance evaluation of OSCAR is the server-side resource signing load. We define parameter β as the mean resigning interval such that $\beta = t/N$, where N is the total number of secured resources on the server and t is the average resource update time (for instance, updates of temperature, pressure, CO_2, etc.). We evaluate OSCAR for β values of 30, 60, and 120 seconds, to account for use cases where high, medium, or low signing load is needed.

In the case of OSCAR, we use pre-shared access secrets and certificates to decrypt and verify encrypted and signed objects. Similarly to the work of Hummen *et al.* [17], we use the secp160r1 elliptic curve. Objects are encrypted using the AES-CCM* algorithm. Similar assumptions apply to DTLS as well: it uses the TLS_PSK_WITH_AES_128_CCM_8 pre-shared key based cipher suite. As a consequence, DTLS only uses symmetric key operations during the handshake.

We have run experiments/emulations over 3 hours and plotted 5 run averages with 95% confidence intervals.

TABLE I
EXPERIMENT SETUP.

(a) WiSMote

Radio Duty Cycling	X-MAC
Channel Check Rate (Hz)	8
Channel Model	Unit Disk Graph

(b) ST GreenNet

Radio Duty Cycling	beacon-enabled 802.15.4
Beacon Interval (ms)	122.88
Superframe Duration (ms)	15.36

Figs. 6(a) and 6(b) show the effect of the reduced radio traffic generated by OSCAR on energy consumption. For medium intensity signing load ($\beta = 60s$), in case of WiSMote, OSCAR crosses the energy performance of compressed DTLS when the client/session slot ratio is approximately 1.3. In case of the ST GreenNet platform, the crossing is increased to approximately 2.15 due to the use of a new generation (prototype) radio with lower consumption. It is important to stress that the exact crossings depend on the consumption characteristics of the MCU and the radio transceiver, and our results are therefore particular for the two evaluated platforms. However, the MCU/radio transceiver combinations on the evaluated platforms are very representative—16-bit CPU and an old generation radio (WiSMote) and a powerful 32-bit CPU with prototype low consumption radio transceiver (ST

GreenNet) allowing us to generalize the crossings between the two.

Although our initial design goal was to relieve constrained servers from radio traffic and to place burden on clients, we can notice in Fig. 7 that even for moderate (in IoT terms) client/session slot ratio (WiSMote 3.7, ST GreenNet 4.17), constant ECDSA verification results in better performance than using the compressed DTLS approach. Note that in our evaluations, we use constrained clients as well thus accounting for the worst case. In IoT use cases, it is expected that a significant part of clients will be more powerful devices such as smartphones, tablets, laptops, or powerful cloud servers.

Finally, we evaluate the request-response latency in Fig. 8. As we can see, MCU computation capabilities greatly affect the result of OSCAR as most of the latency comes from ECDSA verification. In the ST GreenNet deployment, we have observed an increased number of failed DTLS handshakes for the largest evaluated network with 16 clients due to the stochastic nature of radio links. Note that DTLS curves exponentially increase with the number of clients, but are expected to saturate for denser networks. The exact saturation point depends on the configuration of the DTLS retransmission mechanism (we have used the default retransmission timeout of 2 seconds).

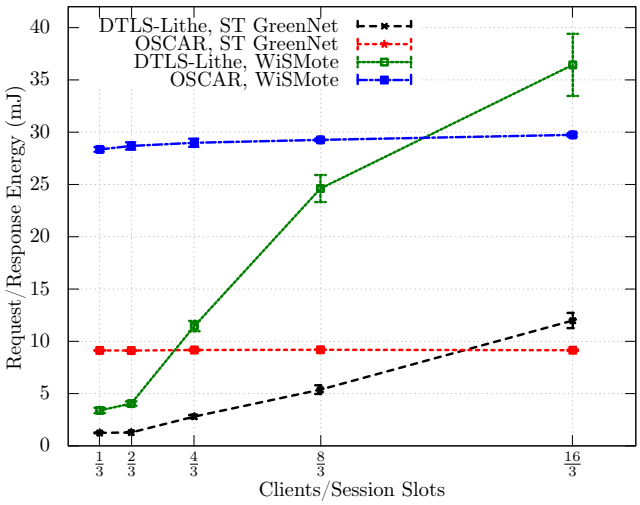


Fig. 7. Client energy consumption per CoAP request-response. It includes a possible DTLS handshake.

VI. Securing the Internet of Things

Research and standardization efforts around secure IoT follow the TCP/IP architectural model by having security features on one (or more) of the layers in the protocol stack. Accordingly, we survey the state of the art.

A. End-to-End Security at the Network Layer

Ever since the efforts on integrating Wireless Sensor Networks with the Internet have begun, the so-called blanket coverage at the network layer has been considered a potential

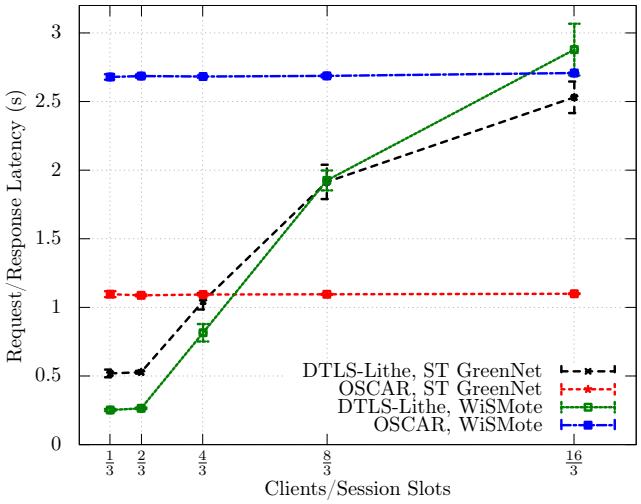


Fig. 8. Request-response latency. It includes a possible DTLS handshake.

solution to provide end-to-end security services [21]. The literature widely discussed the feasibility of porting the IPsec protocol suite to smart objects [22], [23], [24], [3], [25], [26]. The authors mostly evaluated the processing overhead and energy requirements of different cryptographic suites used by IPsec, but also the memory footprints and system response time [3], [25]. Even though it was initially considered too heavy for constrained environments, these results led to the common conclusion that a lightweight version of IPsec is a feasible option.

In the Internet, IPsec mostly secures Virtual Private Networks (VPN). Being at the Network layer, it is perfectly suited for such applications where "blanket" coverage is actually desirable (enterprise networks for example). However, as it resides in the Operating System kernel, it is impractical for typical IoT applications. The requirement that an end user needs to configure the host Operating System and IPsec for securing communication with smart objects would probably result in questionable security practices. Moreover, integrity at the Network layer would prevent any protocol mappings. Namely, as the IP payload is being authenticated, there would be no way of performing HTTP/CoAP mapping at the network gateway. CoAP, however, has been designed from the very beginning to facilitate this for legacy hosts in the Internet.

B. End-to-End Security at the Transport Layer

Impracticality of IPsec has been overcome in the Internet by introducing the security services just below the application layer, in the form of TLS/SSL. The wide and successful use of this model in the Web has also suggested its use in IoT. Indeed, the first proposal on using SSL for smart objects, nicknamed Sizzle, came in 2005 from Sun Microsystems [1]. The authors evaluated the HTTPS stack that leverages assembly optimized implementation of ECC as a public key algorithm. At the time of the publication, however, there was no common agreement on the transport protocol to use. Consequently, the authors

implemented their own reliable transport protocol. SNAIL [27] complemented this work by introducing SSL on an all IP architecture, leveraging the 6LoWPAN adaptation efforts done in the meantime. Together with the introduction of IP to the embedded world came the dilemma whether TCP is suited or not, due to its connection establishment overhead, poor performance in case of lossy networks and short term connections. For this reason, latest standardization efforts [5] assume User Datagram Protocol (UDP) at the transport layer, leaving reliability as an option to the application.

Unreliable transport and possible out of order delivery make TLS as is, an improper candidate for IoT. For the reason of securing application level protocols running over UDP in the Internet, such as Session Initiation Protocol (SIP), Real Time Protocol (RTP), or Media Gateway Control Protocol (MGCP), TLS has already been extended to Datagram TLS (DTLS) [28], [10], which introduced additional 8 bytes of per datagram overhead in the form of the sequence and epoch numbers that were implicitly known with the reliable transport.

As a straightforward and standardized parallel to the successful model in the Internet, DTLS has attracted attention of the research community around the Internet of Things [4], [29], [30], [26], [2], [17]. It is interesting to note, however, that apart from the known advantage of using an already standardized protocol, no argument has been given on actual applicability of DTLS for IoT. Kothmayr *et al.* [30] discussed the necessity of authenticating both the client and the server during the DTLS handshake, but their experimental results show significant completion delays, ranging from 2 to 6.5 seconds. Granjal *et al.* [26] performed a comparative study on memory footprints, computational time, and required energy between IPsec protocols and DTLS, using different cryptographic suites. These results showed similar performance of the two approaches, except in the case when DTLS is additionally used to exchange keys with the Elliptic curve Diffie-Hellman exchange.

Recognizing the excessive overhead of the DTLS handshake, Hummen *et al.* [17] proposed different techniques to lower its impact on constrained devices—certificate prevalidation and handshake delegation to the network gateway. On the other hand, Raza *et al.* tackled the same problem by proposing a 6LoWPAN DTLS compression scheme [4] that reduces per datagram overhead. This work has lately been integrated with CoAP and released in the open source form [2].

A significant drawback of using DTLS to secure IoT is its incompatibility with multicast traffic. As stated by its designers [10], DTLS targets typical connection-oriented client-server architectures. While some of the IoT envisioned applications could loosely undergo this assumption, the majority cannot (cf. Section II). In fact, group communication support is one of the main features why CoAP protocol is being standardized at all [5].

Additional concern raised by the straightforward, point-to-point use of DTLS is incompatibility with scenarios where the end-host in the Internet only supports HTTP/TLS. As stated previously, CoAP has been designed from the beginning to provide easy mapping to HTTP. Brachmann *et al.* [29] discussed a possible DTLS/TLS mapping done at the gateway that preserves E2E security. While verifying integrity at the transport layer, however, it is impossible to perform the CoAP/HTTP mapping at the application layer.

C. Object Security Approaches

Although the concept of object security, *i.e.* placing security within the application payload, has been discussed as an option [5], [31] the related work in the literature leverages its benefits to provide fine grained access control with an assertion-based authorization framework [32]. We jointly address the problems of E2E security and authorization for IoT and use the capability-based access control solely as a means to provide communication confidentiality. The work of Seitz *et al.* [32] is complementary to ours and the two approaches can be merged to cover the use cases where simply responding with an access-protected resource representation is undesired.

D. Standardization Efforts

Recent IETF efforts are directed towards profiling DTLS specifically for constrained devices (DICE working group). Current proposals aim at adding multicast support to DTLS by reusing the record layer and relying on an independent group key management protocol [15]. In essence, the core (D)TLS design assumption (point-to-point communication) is being revisited to make it fit better the IoT requirements.

Authorization and authentication challenges for constrained environments are being tackled separately within the ACE working group. Requirements that are discussed by ACE, however, seem to be contradictory with the initial choice of DTLS as a security protocol, particularly when it comes to proxies and caching. OSCAR bridges this gap and jointly tackles the problems of E2E security and authorization, while keeping full compatibility with the plain DTLS approach.

On the other hand, 6TiSCH working group of IETF designs a security architecture to enable bootstrapping of IEEE 802.15.4 nodes. The main challenges include initial network access and the setup of L2 keys using existing IP protocols.

Finally, it is important to note that different standards specifying the object security format already exist or are under standardization (Cryptographic Message Syntax—CMS, JSON Object Signing and Encryption—JOSE), but their adaptation for constrained devices is required.

VII. Conclusion

Our work explores a novel approach to the problem of E2E security in IoT. It is based on the concept of object security that introduces security within the application payload. We consider separate confidentiality and authenticity trust domains. Confidentiality is used as a means to provide capability-based access control and a protection against eavesdropping during the communication. We protect from replay attacks by coupling the content encryption key with the duplicate detection mechanism of CoAP. Authenticity is tied to the

host and the content encapsulated within objects is digitally signed, which allows the trust in the information to persist long after the actual communication has taken place. In turn, this property enables local databases and caches to use the intrinsically secure content. Moreover, leveraging the access right confidentiality domain and the concept of object security, our proposal intrinsically supports multicast. We take off the burden of a security handshake with every client from constrained servers. Instead, we rely on secure communication channels with Authorization Servers that are in charge of resource access right key management. Cryptographic burden is then shifted to clients that need to perform signature verifications for the content they are interested in.

We have demonstrated the feasibility of the concept by evaluating the proposal in the M2M communication scenario where all parties are resource constrained. We believe that billions of smart, but constrained objects, encompassing IoT are the best argument for a scalable solution such as OSCAR. OSCAR is particularly useful in Smart City deployments where energy constrained servers are expected to have a large number of clients.

We are aware of the fact that the work in this paper may not meet requirements for each and every use case of the vast IoT domain. More specifically, the use cases that require streaming are part of an on-going work.

Acknowledgments

We would like to thank Shahid Raza, Hossein Shafagh and Simon Duquennoy for releasing the implementation of Lithe in open sourced form. Many thanks to Michel Courbon for performing tests on real WiSMote nodes and to Michel Favre for suggestions on porting the Energest benchmarking tool to ST GreenNet platform. The work of F. Rousseau and A. Duda was partially supported by the French National Research Agency (ANR) project project IRIS under contract ANR-11-INFR-016 and the European Commission FP7 project CALIPSO under contract 288879. The work reflects only the authors views; the European Community is not liable for any use that may be made of the information contained herein.

References

[1] V. Gupta, M. Wurm, Y. Zhu, M. Millard, S. Fung, N. Gura, H. Eberle, and S. Chang Shantz, "Sizzle: A standards-based end-to-end security architecture for the embedded internet," *Pervasive and Mobile Computing*, vol. 1, no. 4, pp. 425–445, 2005.

[2] S. Raza, H. Shafagh, K. Hewage, R. Hummen, and T. Voigt, "Lithe: Lightweight Secure CoAP for the Internet of Things," *Sensors Journal, IEEE*, vol. 13, no. 10, pp. 3711–3720, 2013.

[3] S. Raza, S. Duquennoy, T. Chung, T. Voigt, U. Roedig *et al.*, "Securing communication in 6LoWPAN with compressed IPsec," in *DCOSS*. IEEE, 2011, pp. 1–8.

[4] S. Raza, D. Trabalza, and T. Voigt, "6LoWPAN Compressed DTLS for CoAP," in *Distributed Computing in Sensor Systems (DCOSS), 2012 IEEE 8th International Conference on*. IEEE, 2012, pp. 287–289.

[5] Z. Shelby, K. Hartke, and C. Bormann, "Constrained Application Protocol (CoAP) draft-ietf-core-coap-18," *IETF work in progress*, 2013.

[6] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, University of California, 2000, pp. 76-106.

[7] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Emerging networking experiments and technologies*. ACM, 2009, pp. 1–12.

[8] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, "A data-oriented (and beyond) network architecture," in *Computer Communication Review*, vol. 37, no. 4. ACM, 2007, pp. 181–192.

[9] D. Smetters and V. Jacobson, "Securing network content," *Relatório Técnico TR-2009-1, Xerox Palo Alto Research Center-PARC*, 2009.

[10] N. Modadugu and E. Rescorla, "The design and implementation of datagram TLS," in *Proceedings of ISOC NDSS*, 2004.

[11] I. Leontiadis, C. Efstratiou, C. Mascolo, and J. Crowcroft, "SenShare: transforming sensor networks into multi-application sensing infrastructures," in *Wireless Sensor Networks*. Springer, 2012, pp. 65–81.

[12] T. ETSI, "102 691 V1. 1.1 Machine-to-Machine communications (M2M)," *Smart Metering Use Cases*, 2011.

[13] N. Tsiftes and A. Dunkels, "A database in every sensor," in *Conference on Embedded Networked Sensor Systems*. ACM, 2011, pp. 316–332.

[14] T. Jung, X.-Y. Li, and Z. Wan, "Privacy Preserving Cloud Data Access With Multi-Authorities," in *Infocom*. IEEE, 2013.

[15] S. Keoh, S. Kumar, O. Garcia-Morchon, E. Dijk, and A. Rahman, "DTLS-based Multicast Security for Low-Power and Lossy Networks (LLNs) draft-keoh-dice-multicast-security-05," *IETF work in progress*, 2014.

[16] L. Veltri, S. Cirani, S. Busanelli, and G. Ferrari, "A Novel Batch-based Group Key Management Protocol Applied to the Internet of Things," *Ad Hoc Networks*, vol. 11, pp. 2724–2737, 2013.

[17] R. Hummen, J. H. Ziegeldorf, H. Shafagh, S. Raza, and K. Wehrle, "Towards Viable Certificate-based Authentication for the Internet of Things," in *Hot Topics on Wireless Network Security and Privacy*. ACM, 2013, pp. 37–42.

[18] K. Hartke, "Practical Issues with Datagram Transport Layer Security in Constrained Environments draft-hartke-dice-practical-issues-00," *IETF work in progress*, 2013.

[19] M. Vučinić, B. Tourancheau, F. Rousseau, A. Duda, L. Damon, and R. Guizzetti, "Energy Cost of Security in an Energy-Harvested IEEE 802.15.4 Wireless Sensor Network," in *MECO*. IEEE, 2014.

[20] M. Vučinić, G. Romaniello, L. Guelorget, B. Tourancheau, F. Rousseau, O. Alphand, A. Duda, and L. Damon, "Topology Construction in RPL Networks over Beacon-Enabled 802.15.4," in *ISCC*. IEEE, 2014.

[21] M. Luk, G. Mezzour, A. Perrig, and V. Gligor, "MiniSec: a secure sensor network communication architecture," in *IPSN*. IEEE, 2007, p. 479.

[22] J. Granjal, R. Silva, E. Monteiro, J. Sa Silva, and F. Boavida, "Why is IPSec a viable option for wireless sensor networks," in *MASS*. IEEE, 2008, pp. 802–807.

[23] R. Roman and J. Lopez, "Integrating wireless sensor networks and the internet: a security analysis," *Internet Research*, vol. 19, no. 2, 2009.

[24] J. Granjal, E. Monteiro, J. Silva *et al.*, "A secure interconnection model for IPv6 enabled wireless sensor networks," *Wireless Days*, 2010.

[25] S. Raza, S. Duquennoy, J. Höglund, U. Roedig, and T. Voigt, "Secure communication for the internet of things a comparison of link-layer security and ipsec for 6lowpan," *Security and Com. Networks*, 2012.

[26] J. Granjal, E. Monteiro, and J. S. Silva, "On the Effectiveness of End-to-End Security for Internet-Integrated Sensing Applications," in *GreenCom*. IEEE, 2012, pp. 87–93.

[27] S. Hong, D. Kim, M. Ha, S. Bae, S. J. Park, W. Jung, and J.-E. Kim, "SNAIL: an IP-based wireless sensor network approach to the internet of things," *Wireless Comm. IEEE*, vol. 17, no. 6, pp. 34–42, 2010.

[28] E. Rescorla and N. Modadugu, "Datagram Transport Layer Security Version 1.2," IETF RFC 4944, 2012.

[29] M. Brachmann, S. L. Keoh, O. Morchon, and S. Kumar, "End-to-End Transport Security in the IP-Based Internet of Things," in *ICCCN*, 2012.

[30] T. Kothmayr, C. Schmitt, W. Hu, M. Brunig, and G. Carle, "A DTLS based end-to-end security architecture for the Internet of Things with two-way authentication," in *LCN*. IEEE, 2012, pp. 956–963.

[31] S. Cirani, G. Ferrari, and L. Veltri, "Enforcing Security Mechanisms in the IP-Based Internet of Things: An Algorithmic Overview," *Algorithms*, vol. 6, no. 2, pp. 197–226, 2013.

[32] L. Seitz, G. Selander, and C. Gehrmann, "Authorization framework for the internet-of-things," in *WoWMoM*. IEEE, 2013, pp. 1–6.