Nonce-based authenticated key establishment over OAuth 2.0 IoT proof-of-possession architecture

Renzo Efrain Navas, Manuel Lagos, Laurent Toutain, Kumaran Vijayasankar

Nonce-based Authenticated Key Establishment over OAuth 2.0 IoT Proof-of-Possession Architecture

Renzo E. Navas, Manuel Lagos, Laurent Toutain
Department of Network, Security and Multimedia
Telecom Bretagne
Cesson-Sevigne, France
{*renzo.navas, manuel.lagos, laurent.toutain*}*@telecom-bretagne.eu*

Kumaran Vijayasankar
WCS R&D
Texas Instruments
Dallas TX, USA
kumaran@ti.com

***Abstract*—The Internet of Things will scale to billions of devices in the next coming years. A secure communication framework is needed to interconnect all these objects, by taking into account their intrinsic constrained in terms of energy, cpu and memory; Several proposals relying on adapting existing well-known and standardized security solutions exist, but we believe there is still a gap for most-constrained nodes to provide fine-grained authorization and secure establishment of fresh cryptographic keys. We propose a mechanism that runs on top of the OAuth Authorization architecture and provides the bootstrapping of fresh authenticated symmetric cryptographic material between previously unknown parties using a nonce-based protocol. We set up an energy measurement platform to evaluate our proposal and compare it with existing work.**

***Keywords*-authorization; authenticated key establishment; oauth; iot; symmetric; nonce; energy;**

I. Introduction

Constrained Nodes and Networks are predominant on the Internet of Things (IoT). Constraints include limited ROM, RAM, energy and bandwidth. Basic security services as Authentication, Authorization and Confidentiality are needed on most of the IoT scenarios. Standard security solutions such as Public Key Infrastructure (PKI), TLS, X.509 Certificates, and in general asymmetric cryptography, are not directly applicable to the IoT scenarios due to the constrained nature of the nodes and networks.

Several works proposing adaptations to existing solutions or creating new ones for bringing the basic security services for the IoT exist [1]. However, not many address the problem for the most constrained nodes and for very-constrained Low-Power Wide-Area networks: energy and bandwidth are very limited and optimizing the bytes to communicate is a priority. For the constrained node asymmetric cryptography is cpu and memory intensive; hence any solution relying on certificates, signatures, Diffie-Hellman-like key agreement, is excluded.

Our proposal defines an OAuth 2.0-based Authentication and Authorization Framework suitable for the most constrained nodes and networks. We optimize the communication and the cryptographic operations at the node. The proposal will allow the establishment of an authenticated symmetric key, with fine-grained authorization permissions associated, between previously unknown parties. We base our work on nonce-based symmetric authenticated key establishment methods [2], Internet Engineering Task Force (IETF)'s OAuth 2.0 proof-of-possession token and architecture [3] [4], and other lightweight IETF's protocols [5]. Our proposal is analyzed in terms of energy consumption at the constrained node and compared with existing solutions.

The rest of this paper is structured as follows: Section II briefly discuss related work on security on IoT and provides the needed background on which we build our solution. Section III describes our proposed solution. Section IV presents some energy considerations and empirical measurements. Finally, Section V offers some final conclusions.

II. Related Work and Background

A. Related Work

An extensive review of existing IoT security solutions is given on [1]. [6] introduces the problem of securing the most constrained IoT nodes from an energy point of view. Establishment of cryptographic material between communicating parties is the building block of any other security service; [7] gives a thorough introduction to the authenticated key establishment problem for IoT and categorizes the possible solutions. Several lightweight key establishment proposal exist. In [8] the authors offer a scalable DTLS-based solution with a trusted-third-party (TTP), symmetric key freshness is loosely assured by a nonce generated at the TTP. A 6LoWPAN solution is presented in [9] for which they offer a formal proof of security; it solves the problem of a node joining a network. [10] offers an Identity-based approach, who leverages several scalability problems, and is based on asymmetric cryptography. EAP-over-CoAP [11] bootstraps key material and deals with node authentication. The authorization problem for IoT, normally involving authentication, has been also studied thoroughly; [12] offers a comprehensive solution based on EAP-methods and timestamps. [13] solves the authentication problem building on top of 3GPP Generic Bootstrapping Architecture (GBA) using asymmetric cryptography and HTTP. IoT OAuth-based authorization

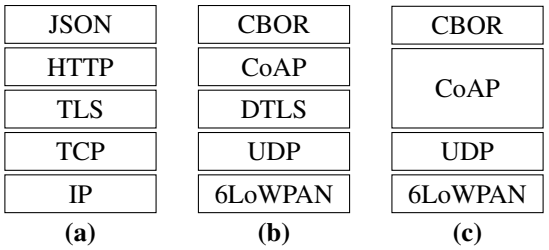

Figure 1. Protocols stack for: (a) Standard OAuth; (b) IoT-profiled OAuth; (c) IoT-profiled OAuth w/PoP Tokens and Object Security.

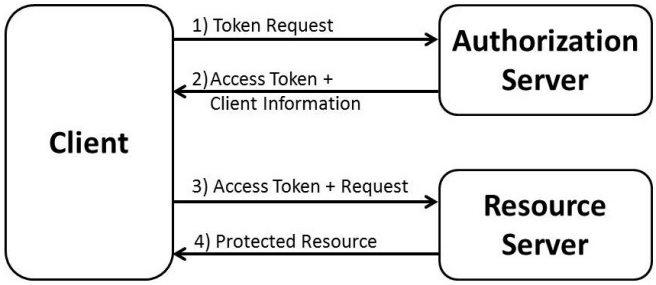

Figure 2. OAuth 2.0 Basic Messages Flow and Entities

solution exists. In [14] they profile OAuth-for IoT using regular tokens, but authenticated key establishment between unknown parties is out of scope. [15] uses also regular tokens that need to be transported securely, and freshness relies on timestamps validation.

B. Authorization: OAuth 2.0 for IoT

The OAuth v2.0 Authorization Framework [16] is an open standard that enables a resource owner to grant a third-party limited access to a protected resource. OAuth 2.0 provides an authorization layer that is designed to be used over HTTP. An authenticated secure channel between communicating parties is needed for most messages and is generally provided by a transport-layer security solution such as TLS.

Access Tokens are authorization credentials that grant access to the protected resources; they can have different formats, cryptographic properties and uses. One example is the bearer token: any party in possession can use it to get access to the associated resources. The *Proof-of-Possession (PoP) Tokens* are access tokens bound to a specific cryptographic key. Mere possession of the PoP token is not enough to access a protected resource: possession of the PoP token's associated key must be proven too. PoP tokens' semantics, distribution and associated architecture are being developed at IETF's OAuth Working Group (WG) [3]. One desirable security property of PoP tokens is that they can be transported over unsecured channels.

IETF ACE's work-in-progress *Authentication and Authorization for Constrained Environments (ACE)* [4] aims at adapting OAuth v2.0 for the IoT. It uses PoP access tokens; HTTP is replaced by CoAP and JSON by CBOR, CoAP runs over UDP instead of TCP, and hence DTLS should be used instead of TLS. Application-layer security solutions are also envisioned and are based on COSE (CBOR Encoded Message Syntax) [5].

The protocols stack and data model representation needed for an entity taking part on the OAuth framework, assuming an IP layer, can be seen in Figure 1.

1) OAuth Main Entities: The main entities involved in the OAuth framework are:

- *Resource Owner (RO)*: An entity that controls the authorization permissions of a protected resource.
- *Resource Server (RS)*: An entity which hosts a protected resource.
- *Authorization Server (AS)*: An entity that enforces RO's policies, prepares and endorses authorization and authentication data.
- *Client (C)*: An entity which attempts to access a protected resource on a RS.

2) OAuth Message Flows: Figure 2 represents the basic OAuth 2.0 message flow. We assume the RO has preconfigured the authorization policies on the AS and it is not relevant on the flow.

The procedure that allows C to get access to a protected resource is the following:

1) C sends a *Token Request* message to the AS.
2) If C is authorized AS generates and sends to C the *Access Token* (opaque to C) and *Client Information* (e.g. contains the key bound to the PoP access token).
3) C sends the *Access Token* to RS and the protected resource Request
4) RS validates the request with the associated access token, if successful, responds with a representation of the protected resource.

C. Authentication: Authenticated Key Establishment (AKE)

Authentication and key establishment protocols are the pillar of secure communications. *Key Authentication* is a property that ensures a party that another identified party is in possession of a particular key. An *Authenticated Key Establishment* (AKE) protocol is a key establishment protocol that provides key authentication. The current most used AKE protocols are the Kerberos v5 and authenticated Diffie-Hellman variants used on TLS cipher suites.

1) AKE and a Trusted Third Party: A relevant statement that has been formally proven to be true [17] is that no fresh authenticated key establishment can be done if there is not already existing secure channels. For two entities that want to establish a fresh authenticated key this means that either they: (a) already share some cryptographic material; (b) share cryptographic material with a mutually trusted

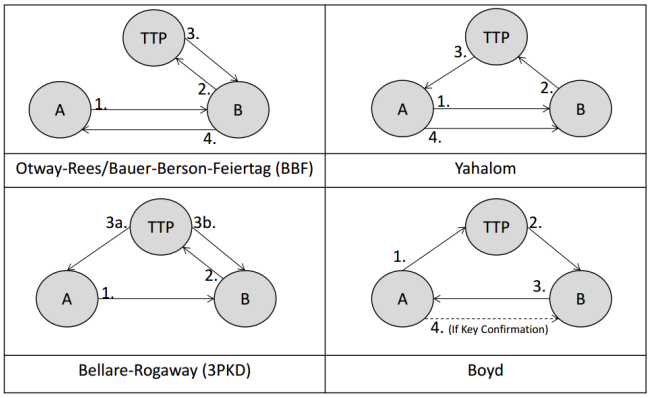

Figure 3. Three-party symmetric nonce-based AKE protocols messages flow. Party A and B establish a key with a TTP involved.

third party (TTP). For parties without shared cryptographic material the AKE establishment will involve a TTP either in an on-line fashion interacting on the protocol as an active party, or in an off-line fashion like in public key infrastructures.

2) AKE: Three-Party, Symmetric and Nonce-based: AKE protocols can be categorized depending on: the number of parties involved; the use of symmetric or asymmetric cryptography; the use of timestamps or nonces (to assure freshness). *Three-party symmetric cryptography nonce-based* AKE protocols will be the basis for our work. Figure 3 shows the message flows of the most relevant protocols in the literature.

3) The Bauer Berson and Freiertag (BBF) protocol: Among the five highlighted AKE protocols we will focus on the the Bauer-Berson-Freiertag's (BBF) [2]. The main reason for its choice is a desired property of the messages flow: *one of the involved parties does not need direct communication with the TTP*. Otway-Rees protocol also has this property but attacks exists. While the BBF protocol has not a provably secure proof of security (3PKD is the only one), it has no known-attacks. To describe BBF we define the following notation: A and B are entities wishing to establish a key; S is the Trusted Third Party (TTP); N_A and N_B are nonces generated by A, and by B respectively; K_{AS} and K_{BS} are the long-term shared keys between A and S, and by B and S respectively; $Enc(m)K_k$ Encryption of message m with the key K_k; K_{AB} is the key to be shared by A and B. The BBF protocol runs as follows:

1 : $A \rightarrow B : A, N_A$
2 : $B \rightarrow S : A, N_A, B, N_B$
3 : $S \rightarrow B : Enc\{K_{AB}, B, N_A\}_{K_{AS}}, Enc\{K_{AB}, A, N_B\}_{K_{BS}}$
4 : $B \rightarrow A : Enc\{K_{AB}, B, N_A\}_{K_{AS}}$

A and B inside the messages are the identities of each party. How to represent them is out of scope on AKE protocols, and it is a field of research on its own; on real deployments we can assume IPv6 addresses or DNS names.

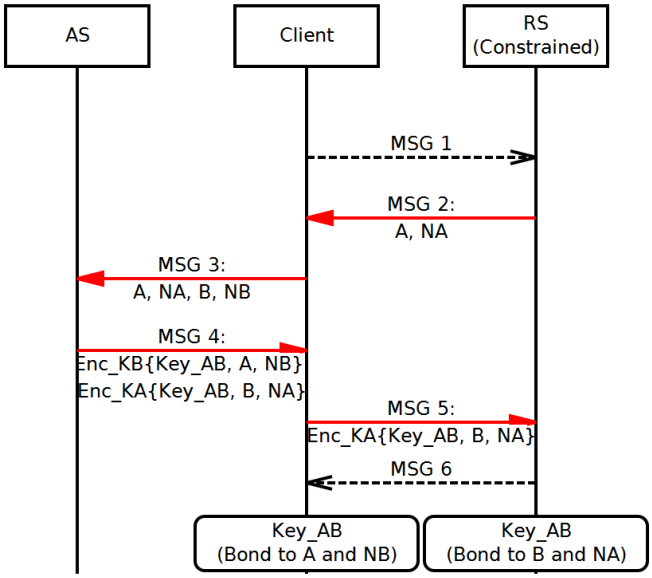

Figure 4. BBF AKE protocol on top of the OAuth Architecture. Authorization Server (AS) act as TTP, $Client$ is party B, and Resource Server (RS) is party A. In solid lines are cryptographic-relevant Messages (KA and KB are long-term shared keys between RS and AS, and by $Client$ and AS respectively).

D. Limitations of OAuth for IoT

The OAuth 2.0 profiled for IoT is a suitable solution for the authorization problem. The PoP Token offers also the secure establishment of an authenticated key -with associated authorization permissions- between previously unknown parties and over an unsecured channel; this key can be further used to establish authenticated and secured communications between these parties. In spite of this advantages we see a mayor drawback on the OAuth PoP token mechanism, its security relies either on: (a) the RS validating a timestamp on the token, or (b) RS directly communicating with AS to delegate the token validation (*token introspection*). We propose a solution that, being based on the BBF AKE protocol, requires neither.

III. Proposal: BBF AKE protocol on OAuth

Our proposal extends the OAuth 2.0 PoP [3] architecture and allows to run the BBF AKE protocol on top of it.

We define new types and flow of messages. The new flow of messages and associated BBF protocol's cryptographic-relevant information is shown in Figure 4; The Client (C) is the initiator of the OAuth protocol and act as party B in the AKE literature, the constrained Resource Server (RS) act as party A (the initiator of the BBF protocol), and the Authorization Server (AS) act as the TTP.

We extend the COSE [5] and CBOR Web Token (CWT) encodings, defining a new type of `key_algorithm` for the `COSE_Key` CBOR object; the object indicates that a BBF

protocol key is being negotiated, and includes an associated context COSE_AKE_Context with the necessary fields to run the BBF protocol. The definition of the CBOR object can be formally described using the CBOR Data Definition Language (CDDL):

```
COSE_Key = {
  key_type       => int / tstr,
  ?key_algorithm => int / tstr,
  ?key_kid       => bstr,
  ?key_operations=> [+ (int) ],
  ?key           => bstr,
  COSE_AKE_Context = [  ?AlgorithmID  : int / tstr,
                        PartyAInfo    : [ PartyInfo ],
                        PartyBInfo    : [ PartyInfo ],
                        ?keyDataLength: uint ]
}
PartyInfo = ( ? nonce: bstr / int, ? identity : bstr / int)
```

The object is composed of field name and value pairs. The field names have a pre-defined binary value called *label* (e.g: key_type label is 1); the field values can be of different types such as: arrays ([]), integer (int), unsigned integer (uint), byte strings (bstr), UTF-8 text string (tstr); and can be optional (?).

We will describe the proposed protocol message's flow and semantics. The focus will be on RS. We assume that RS and AS share a long-term key K_A, and the $Client$ (C) and AS share the key K_B. A successful run will be as follows:

1) C sends to RS an unauthorized CoAP Request (*MSG 1* in Figure 4).
2) RS generates a nonce N_A and together with a representation of its identity A sends a COSE_Key with the BBF information on the response (*MSG 2*). We show the payload in a JSON-like notation:

```
Header: Unauthorized (Code=4.02)
Content-Format: "application/cose-key"
Payload:<
{
  key_kty   : Symmetric,
  key_alg   : AKE-BBF,
  key_kid   : 0x01,
  key_ops   : deriveKey,
  COSE_AKE_Context :
  {
    AlgorithmID : AES-CCM,
    PartyAInfo  :{
       nonce    : N_A, /* 128 bits */
       identity : "a.rserver.domain"
     },
     keyDataLength : 128
  }
}
>
```

3) *Token Request*. C adds a nonce N_B and a representation of its identity B to the COSE_Key. C sends to AS a *COSE-Encrypted CBOR Token Request* containing the COSE_Key (*MSG 3*). This message has to be authenticated (and encrypted if sensitive client credentials are sent) because AS will enforce authorization policies regarding C.
4) *Token Response*. AS replies to C with a *COSE-Encrypted CBOR Token Response* (*MSG 4*). This response contains the *PoP Access Token* that is opaque to C, and *Client Information* (CI). The most relevant CI is the COSE_Key that contains: the new symmetric key K_{AB}, the nonce N_B and A identity representation; these three values are encrypted from AS to C.
5) *Token Presentation*. C sends the PoP Access Token to RS (*MSG 5*). This is done with a CoAP POST to the /authz-info resource at RS. This message does not need to be secured by C as the PoP Token is already protected between AS and RS. The content of the CoAP message will be the following, we include also a successfully decrypted message by RS that will contain the plaintext CBOR Web Token with a COSE_Key (on the ck field):

```
Header: POST (Code=0.02)
Uri-Path:"authz-info"
Content-Format: "application/cwt;"
Payload: <COSE-Encrypted CBOR Web Token>
Payload Plaintext:
{
  aud : "a.rserver.domain",
  aif :  [["/r", 0], ["/other", 2]],
  cnf : {
     ck:
     {
        key_kty : symmetric
        key_kid : 0x01,
        key_alg : AES-CCM-16-64-128,
        key      : K_AB, /* 128 bits */
        COSE_AKE_Context:
        {
            PartyAInfo :{
                nonce    : N_A, /* 128 bits */
            },
            PartyBInfo :{
                identity : "b.client.domain"
            }
        }
     }
  }
}
```

As per BBK AKE properties, C will associate the K_{AB} with identity B and N_A, which guarantees key's authentication and freshness. OAuth associated authorization permissions are in aif. Notice that the key_alg does not contain the value AKE_BBF but is now the algorithm that should be used for authenticated encryption, in this case AES-CCM mode 128-bit key, 64-bit tag, 13-byte nonce.
6) If RS correctly validated the last message it sends to C an ACK 2.04 Changed response (*MSG 6*)

After this exchange of messages C and RS, who previously shared no cryptographic material, will share a symmetric key K_{AB}. This key's freshness and security properties are assured by the BBF AKE protocol; and is bound to an OAuth's *Proof-of-Posession Access Token* who grants to the holder-of-key fine-grained authorization permissions to access protected resources on RS. This K_{AB} can be further used to set-up an authenticated and secured channel between C and RS by any method that can do so using a Pre-Shared-

Key (PSK), such as DTLS-PSK or Object Security (COSE).

IV. Considerations and Experimental Results

A. Design Considerations

We designed the protocol minimizing the resource consumption on RS, from a high level point of view this can be achieved minimizing:

- The number of bytes to communicate to and from other nodes (Bytes TX/RX. Energy).
- The cryptographic operations needed, qualitatively and quantitatively (CPU Power, RAM).
- ROM and RAM usage, by using an optimized protocol stack and information encoding schemes.

We aim at energy constrained nodes, battery-powered typically with a CR2032 coin cell with a capacity of 225mAh.

B. Experimental Results

1) Test method: Texas Instruments (TI) CC1310 was used as a constrained node. RAM and ROM usage were not measured empirically, focus was put on energy. We set up an energy-measurement platform using: (a) TI CC1310 Evaluation Module (EM): CPU ARM Cortex-M3 (@48 MHz), 128 KB of ROM, 20KB RAM, Sub-1 GHz Low Power Radio, AES-128 Security Module and a True Random Number Generator (TRNG) module. (b) TI SmartRF06 Evaluation Board to program and debug the TI CC1310 EM, and to measure current consumption (voltage drop ΔV on a shunt resistor). (c) Tektronix TDS 3012 Digital Oscilloscope (100 MHz, 1.25 GS/s). We powered the Evaluation Board (EB) from the USB 5.0V source, the EB has a DC-DC converter to 3.30V. We used two different values for the shunt resistor used to measure ΔV: 5Ω and 22Ω. We also used an output digital pin as a trigger for the oscilloscope to capture the desired operation.

Figure 5 illustrates the method used for measuring each operation, in this case the TRNG: it took 472 msec to generate a random number; the CPU enters a low-power mode while the TRNG gathers entropy, this explains the initial reduction in power consumption.

2) Results: We measured the following operations in the CC1310 EM: (a) TX: Sending 25 and 50 bytes over the radio. (b) RX: Receiving 25 and 100 bytes over the radio. (c) Generating a True Random Number with the TRNG module (d) Normal operation CPU consumption. The calculated current consumption and elapsed time for each operation is summarized in Table I.

We can estimate the consumption of RS, in terms of mAh, to complete the protocol. We assume a packet overhead from Layer 2 to CoAP of 20B. Using Figure 4 as reference:

- *MSG 1*: RX 25 bytes (6.3 mA x 5.8 msec)
- *MSG 2*: Generation of a TRNG (3.0mA x 474.0msec) + TX 50 bytes (27.1mA x 9.9msec)
- *MSG 5*: RX 100 Bytes (6.3mA x 18.6msec) + AES-CCM Decryption of 80 Bytes ((0.5 + 3.2) mA x 1msec)

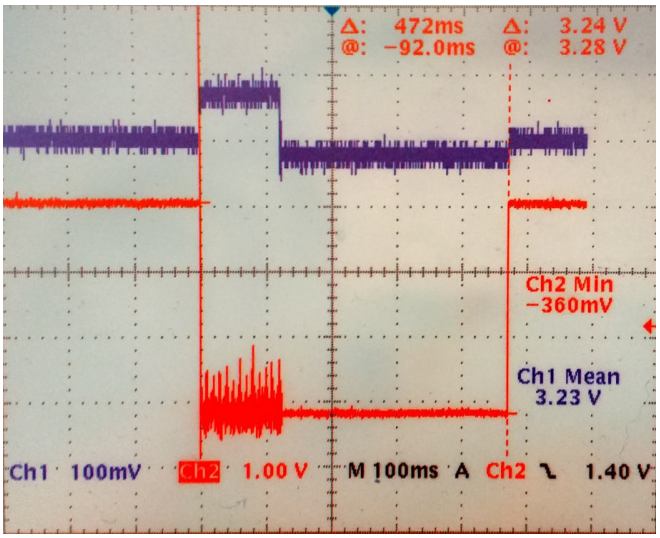

Figure 5. Measured Δtime = 472 ms (Δt_1: 132ms@3.30V; Δt_2: 342ms@3.21V) and Δvoltage over a 22 Ω shunt resistor for a generation of a True Random Number with the TRNG module in TI's CC1310. Lower part is the trigger signal to indicate the duration of the operation. Top is the voltage drop over the shunt resistor. Oscilloscope Tektronix TDS 3012.

Table I
TI CC1310 Current consumption and time execution of relevant operations to run the proposed protocol

Operation	Mean Current (mA)	Time Elapsed (msec)
TX	27.1	5.8 [25 B] / 9.9 [50 B]
RX	6.3	5.8 [25 B] / 18.6 [100 B]
TRNG	3.0	474.0 [16 B]
AES Module	0.5*	1.0*[CCM-Decrypt 80 B]
CPU Active	3.2 *[Adds to AES module]*	-

* value taken from datasheet.

- *MSG 6*: TX 25 Bytes (27.1mA x 5.8msec)

The sum of the terms gives a total of **0,000557mAh**. At 3.30V this is equivalent to **6.62 mJoules**.

The proportional contribution of each operation to energy-consumption are: (1) TRNG: 71%. (2) Radio TX: 21%. (3) Radio RX: 8%. (4) AES operations $\approx$0%.

If we use a CR2032 battery (225mAh) only for running the protocol we can do it in the order of 400 000 times before battery exhaustion. An application needing one protocol run per day and expected to run 15 years on the same CR2032 battery will use 1.4% of its capacity.

3) Axes for improvement: Avoiding the TRNG (71% of energy). We can generate one true randon number N_1 only at the first boot, once per node lifetime, and store it in flash/rom together with a counter n; then for each nonce N_n needed: $N_n = \text{AES-Encryption}(N_1 + n)_{K_{non}}$. Being K_{non} a key for this specific use and incrementing and storing n value. This improvement will give the protocol consumption of 0,000163mAh / 1,94 mJ@3.30V. Minimizing TX and RX; using Figure 4 as reference: *MSG 6* is not related to security and can be avoided, giving a consumption of **0,000119mAh**

Table II
ENERGY CONSUMPTION OF PROPOSALS ON THE LITERATURE

Proposal	Hardware	Energy (mJ)
Saied et al. [7]	TelosB	60 (TLS) / 40 (IKE)
Raza et al. [8]	TI CC2538*	1 / 0.2 (w/HW crypto.)
Porambage et al. [10]	TelosB	1.5
Sethi et. al [13]	Arduino MEGA	70
Ciriani et. al [14]	MSP430+CC2420*	1.3 (IP) / 1.7 (IPsec)
Proposed Solution	TI CC1310	6.6 / 1.4 (improved)

* current value taken from datasheet.

/ 1,42mJ@3.30V. New distribution: TX 62%; RX 36%; AES 2%; TRNG 0%. We can further optimize TX/RX: in *MSG 2* RS's identity is not needed, C already contacted it; in *MSG 5* we can have some fields implicit.

4) Comparison to other solutions: Table II lists energy consumption from the studied literature: our improved solution is in the order of 1.5 mJ, in line with the lowest-energy solutions. A more accurate empirical comparison should be done on the same hardware platform.

C. Perspectives

We envision having a complete implementation of the proposed framework running in constrained nodes and using a desktop PC as an Authorization Server. We are interested in doing energy measurements in different radio modes (e.g. low-energy), and RAM and ROM requirements. On the architectural point of view, we aim at having a flexible framework in which implementing other types of AKE protocols, like 3PKD, will be straightforward; then we can benchmark them and use the appropriate one depending on the constraints of the scenario envisioned. Securing and optimizing the application data exchange is also on scope.

V. CONCLUSION

Fine-grained dynamic authorization is a fundamental service in the IoT world to come. Establishment of authenticated cryptographic key material between previously unknown parties is also the pillar for secure communications. Our proposal is an effort towards a standardized solution that tackles both security problems. We have focused on the most constrained devices, particularly assuming constraints in terms of energy and the impossibility of time-based solutions to assure freshness.

ACKNOWLEDGMENT

The authors would like to thank Nicolas Montavont for his helpful recommendations; Ariton Xhafa, Dominique Poissonnier and Benjamin Cama.

REFERENCES

[1] J. Granjal *et al.*, "Security for the Internet of Things : A Survey of Existing Protocols and Open Research issues," *Communications Surveys & Tutorials, IEEE*, 2015.

[2] R. K. Bauer, T. A. Berson, and R. J. Feiertag, "A key distribution protocol using event markers," *ACM Transactions on Computer Systems*, vol. 1, no. 3, pp. 249–255, 1983.

[3] *OAuth 2.0 Proof-of-Possession (PoP) Security Architecture*. Internet-Draft, IETF. [Online]. Available: https://tools.ietf.org/html/draft-ietf-oauth-pop-architecture-07

[4] E. Wahlstroem *et al.*, "Authentication and Authorization for Constrained Environments (ACE)," IETF, Internet-Draft draft-ietf-ace-oauth-authz-02, Jun. 2016, work in Progress.

[5] J. Schaad, "CBOR Encoded Message Syntax," IETF, Internet-Draft, 2016, work in Progress. [Online]. Available: https://tools.ietf.org/html/draft-ietf-cose-msg-12

[6] W. Trappe, R. Howard, and R. S. Moore, "Low-Energy Security : Limits and Opportunities in the IoT," *IEEE Security and Privacy*, vol. 13, no. 1, pp. 14–21, 2015.

[7] Y. B. Saied, A. Olivereau, D. Zeghlache, and M. Laurent, "Lightweight collaborative key establishment scheme for the Internet of Things," *Computer Networks*, 2014.

[8] S. Raza, L. Seitz, D. Sitenkov, and G. Selander, "S3K : Scalable Security With Symmetric Keys DTLS Key Establishment for the Internet of Things," *IEEE Transactions on Automation Science and Engineering*, pp. 1–11, 2015.

[9] Y. Qiu and M. Ma, "An authentication and key establishment scheme to enhance security for M2M in 6LoWPANs," *2015 IEEE International Conference on Communication Workshop, ICCW 2015*, pp. 2671–2676, 2015.

[10] P. Porambage, A. Braeken, P. Kumar, A. Gurtov, and M. Ylianttila, "Efficient Key Establishment for Constrained IoT Devices with Collaborative HIP-based Approach," *IEEE Global Communications Conference*, pp. 1–6, 2015.

[11] D. Garcia, R. S. Sanchez, and R. Lopez, "EAP-based Authentication Service for CoAP," IETF, Internet-Draft, Apr. 2016, work in Progress. [Online]. Available: https://tools.ietf.org/html/draft-marin-ace-wg-coap-eap-03

[12] J. L. Hernandez-Ramos, M. P. Pawlowski, A. J. Jara, A. F. Skarmeta, and L. Ladid, "Toward a lightweight authentication and authorization framework for smart objects," *IEEE Journal on Selected Areas in Communications*, vol. 33, 2015.

[13] M. Sethi, P. Kortoc, M. D. Francesco, and T. Aura, "Secure and Low-Power Authentication for Resource-Constrained Devices," *5th International Conference on the IoT*, 2015.

[14] S. Cirani *et al.*, "IoT-OAS: An oauth-based authorization service architecture for secure services in IoT scenarios," *IEEE Sensors Journal*, vol. 15, no. 2, 2015.

[15] S. Emerson *et al.*, "An OAuth based Authentication Mechanism for IoT Networks," in *Information and Communication Technology Convergence (ICTC), Internat. Conf. on*, 2015.

[16] E. D. Hardt, *The OAuth 2.0 Authorization Framework*. RFC 6749, IETF, 2012. [Online]. Available: https://tools.ietf.org/html/rfc6749

[17] C. Boyd, "Security architectures using formal methods," *IEEE Journal on Selected Areas in Communications*, vol. 11, no. 5, pp. 694–701, 1993.