An IdM and Key-based Authentication Method for providing Single Sign-On in IoT

Adriano Witkovski, Altair Santin, Vilmar Abreu, João Marynowski
Graduate Program in Computer Science
Pontifical Catholic University of Parana
Curitiba, Parana, Brazil

Abstract—**Internet of Things (IoT) brings significant challenges to authentication schemes in a scenario with several appliances for a smart house that should be accessed by a technician for maintenance tasks, for instance. An Identity Management (IdM) can be applied in order to easily authenticate a technician that intend to access the appliances from the Internet. However, Internet context is significantly different from IoT, demanding context adaptation to work. Thus, integrate these contexts to allow the authentication on the Internet and provide Single Sign-On (SSO) in IoT is a challenge. The goal is to allow a technician to access an appliance that is not reachable from the Internet, using IdM and without create a single compromising point - a critical entity for security - in the gateway that link the two contexts. The proposal interact two key-based scheme, one for Internet and another for IoT to reach integration between both contexts. A proof-of-concept implementation shows the proposal is feasible and did not affect the message exchanging with up to 1024 bytes and 50 appliances.**

Keywords—Internet of Things; Identity Management; Key-based Authentication; Single Sign-On; Smart House

I. Introduction

Internet of Things (IoT) is a network with multi-interconnected objects called "things". Things may be tags, actuators, sensors, microprocessors, and appliances, which are identifiable by a unique network address and they have connectivity to interact with each other, manufacturer and users. Usually, things have resource constraints, such as low processing, memory and communication (bandwidth and range) [1].

IoT is present in several areas such as health, transportation, automation and residential. IoT can be present in a residential scenario where several appliances (e.g. refrigerators and washers) can be configured to monitor people habits or to assist them with daily life tasks, e.g. informing a manufacturer about an appliance malfunction. In such a case, manufacturers need to access appliances remotely, either to repair and optimize or to update its firmware. The obliquity and interconnection of multiple devices make important the appliances isolation, i.e., they are not directly accessible from the Internet to prevent unauthorized access and privacy violation. Other security requirements need to be guaranteed in the IoT, as data secure communication, application secure access, and Identity Management [2, 3].

Authentication and access authorization are significant challenges for the IoT because, different from the traditional Internet components, the appliances are based on specific purpose devices, usually with constrained resources. An Identity Management (IdM) system provides authentication and access authorization for Internet users [4]. However, integrating an Internet IdM with the IoT is not trivial, due to the appliances resource constraints and the lack of communication security between Internet context (e.g. stack of protocols, authentication mechanisms and exchanges, and public key infrastructure) and IoT contexts (characterized by devices for specific purposes and predominantly with resource constraints). Thus, integrate authentication and access authorization from the Internet to the IoT appliances, focusing on security and using current technology, is a major challenge.

The authentication and access authorization approaches for IoT typically use a single password for all appliances or an authentication service. A single password has the advantage of requiring fewer resources, but someone can get access to any appliance once discovered a password, and password updating in all appliances is not a trivial task [3]. An authentication service allows to solve the shortcomings of the single password approach. Some proposals tried to integrate Internet and IoT contexts, but they lack implementation, require appliances adaptations that make infeasible its adoption or simplify the mechanisms in a way that security became ineffective [11, 12, 13]. Furthermore, no approach deals with end-to-end security channels and Single Sign-On (SSO), for temporary access to various appliances from a single authentication.

Our hypothesis is that it is possible to integrate an Internet IdM with the IoT, considering the appliances resources constraint, without exposing the gateway security, as single point of compromising. For this purpose, we encrypted the data content using a symmetric key in an end-to-end communication between appliance and manufacturer. Additionally, to provide per-message protection, we use a secure channel communication between appliance and gateway, and between gateway and manufacturer.

The proposal inherits SSO from IdM, in the Internet context, between technician and gateway and allows a key-based SSO for IoT, between technician and appliance. This scheme is feasible because manufacturer's technician and gateway share the same authentication and access authorization server, and appliance and manufacturer's technician share the same symmetric key server, the Customer Service. The proposed scheme is based on IT standards, and the prototype uses well-known technologies for Internet and IoT integration.

The paper is organized as follows. Section 2 shows the fundamentals of IdM and key management. Section 3 addresses related work. Section 4 presents the proposal. Section 5 shows prototype and evaluation, and Section 6 draws conclusions.

II. Fundaments

Identity Management (IdM) aims at providing authentication and access authorization for users in the Internet environment [4]. It has four components: entity (users or devices), identity (entities identifiers), Identity Provider (IdP) and Service Provider (SP). An IdP manages the users' identities and the authentication attributes, providing credentials for the access authorization. A credential comprehends the entities and actions a user is authorized to access. An SP provides services to authorized users according to their identity and credentials. An example of IdP is OpenID Connect [5]. It allows to integrate authentication and access authorization in a way that an application does not need to manage the identities, passwords, and access authorizations of its users. OpenId Connect is an identity management system that uses the OAuth protocol [5] for the access authorization.

An IdM system also allows deploying Single Sign-On (SSO) service [5]. This service allows a user to authenticate once in a period, and its authentication credentials remain valid for the usage in several SP. Thus, each SP validates the identity and credentials with an IdP, preventing the user needs to enter its identifier and password to access each service or resource. However, an IdM is designed for Internet environment, where the components do not have constrained resources. Thus, integrate IdM to IoT context remains a challenge.

The IoT protocols stack is suitable for the appliances constrained resources. The IEEE 802.15.4 standard is used in the physical layer since it allows wireless communication with low power consumption, but it has low range and low transmission rates [6]. The IPv6 Over Low Power Wireless Personal Area Networks (6LoWPAN) is the protocol used in the network layer since it applies compression and encapsulation mechanisms, and it allows receiving and sending Internet packages (IPv6) using IEEE 802.15.4 [6]. The User Datagram Protocol (UDP) is used in the transport layer due to the overhead of the Transmission Control Protocol (TCP), commonly used on the Internet.

Appliances communicate using Constrained Application Protocol (CoAP). CoAP is based on the Representational State Transfer (REST) architecture, where the resources controlled by a server are identified and accessed by the Uniform Resource Identifier (URI) [7]. CoAP uses UDP in the transport layer, and it can use the Datagram Transport Layer Security (DTLS) to provide per-message protection, including integrity and confidentiality. CoAP integrated with DTLS is also called CoAPs [8].

The ANSI X.9.17 [9] is a standard for managing the symmetric key encryption. The standard defines a scheme to distribute keys, establishing a three-level hierarchy between key pairs. The highest level uses Master Key Encrypting Key (KKM), which is distributed offline and manually between pairs. The intermediate level uses Key Encrypting Key (KEK), which is distributed online, during communication. The lower level uses Key Data (KD), which is also distributed online and encrypted the communication data. KEK and KD are periodically changed and encrypted with KKM.

III. Related Works

Liu et al. [10] propose an authentication architecture and access control for IoT devices and users. In the proposal, devices are considered final nodes of the Internet architecture and can communicate directly through global unique addresses, using IPv6. For authentication and authorization, the authors propose to use OpenID and Role-Based Access Control (RBAC), respectively. The proposal does not address SSO issues and does not present results that can validate the proposal.

Thuan et al. [11] propose an user-centered identity management, integrating IoT to Internet. The appliances access control is performed by an external authentication mechanism, using an IoT identification structure. The proposal aims to use IdM in the IoT without considering security aspects. Fremantle et al. [12] propose to control access to the appliances through the OAuth protocol and a protocol based on message queue for an intermediary component between IoT and Internet contexts. Battisti et al. [13] proposed a federated architecture model in the context of the smart house. The authors propose to use an intermediary Web Services component between Internet and IoT, providing messages security by the WS-Security (Web Services Security), aiming at messages integrity and confidentiality. The proposals [11, 12, 13] do not consider end-to-end security interaction between IoT and Internet, secure channels, and neither SSO.

Cirani et al. [14] present an architecture for an external authorization service based on the OAuth, called IoT-OAS. The proposal addresses the integration of IoT with an Internet authorization scheme using a secure communication channel between pairs. However, the work does not have the end-to-end security integration between Internet and IoT, allowing the intermediary to be a single point of security compromising. Moreover, it does not address SSO.

Chibelushi et al. [15] proposed an IdM system for IoT considering health context. However, the proposal focuses on using Mobile Ad Hoc Networks (MANETs) and does not provide secure communication, exposing all devices directly to the Internet.

Other proposals aim to provide authentication in different ways. Hummen et al. [16] present an authentication method based on certificates, using DTLS and aiming at performance and communication overhead reduction. The proposal does not consider SSO in the IoT context. Li et al. [17] propose to use Lightweight Directory Access Protocol (LDAP) and Kerberos to provide authentication and SSO in IoT. However, the proposal does not consider an intermediary component to adapt the Internet messages to IoT. Yao et al. [18] present a lightweight mechanism for multicast authentication on a small scale IoT, but they did not also address SSO.

Although several works proposed to use an IdM in IoT and other authentication and access authorization schemes, several issues remain open, as a feasible scheme to integrate both contexts. Some proposals do not consider security aspects, such as secure communication channels, the use of IoT protocol stack, and symmetric keys encryption. Other proposals aim to integrate Internet and IoT contexts but do not consider end-to-end security integration, allowing single points for security compromising, as exploiting a vulnerability to intercept and manipulate messages. Many proposals do not apply SSO, and several did not show experimental results, hindering their viability evaluation.

IV. Proposal

In this section, we present a key-based authentication and access authorization scheme for end-to-end security in IoT. Our approach securely integrates an IdM with IoT, considering the appliances resource constraints and providing SSO. The gateway between Internet and IoT is not characterized as critical from the security viewpoint since the communication occurs encrypted symmetrically between the end parties, preventing the interception and manipulation of message contents. In the following sections, we present the proposed scheme in more details, starting with the architecture overview, and after the messages exchange between proposal components.

A. Overview

The proposal involves six components (Figure 1): Appliance, Customer Service, Gateway, Appliance Technician, Authentication Server, and Access Authorization Server. Appliance (App) is a "thing" of the IoT used in a smart house, for instance, which has limited resources and without direct access to the Internet. We aim at preserving privacy and unauthorized App access by making it inaccessible directly from the Internet. App has an identifier attribute (e.g. serial number) and a symmetric key, provided by the manufacturer during its assembly line production. The symmetric key and serial number are stored in the Customer Service. Customer Service (CS) is a service provided by the manufacturer that performs the communication interface between App and a technician. A technician responds requested demands, such as App activation, monitoring, maintenance and firmware upgrade. Gateway (GW) is responsible for linking Internet to IoT, enabling the messages exchange between App and CS.

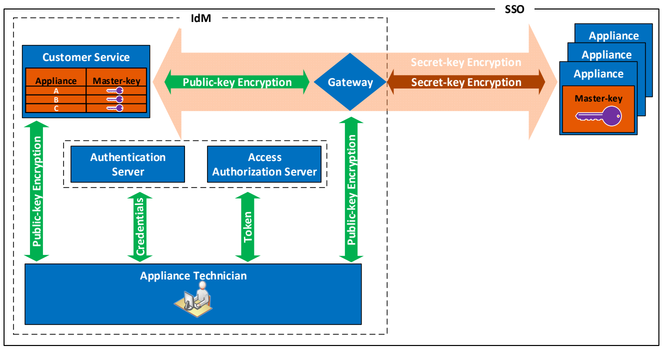

Figure 1. Overview of the IdM and key-based authentication scheme to provide SSO in IoT.

Appliance Technician (AppTec) is a system operated by a manufacturer technician to respond consumer demands and App after-sales needs. AppTec does not have direct access to the App; it uses CS as a bridging element for this. Authentication Server (AS) and Access Authorization Server (AAS) compose an IdP. AS provides authentication service that validates AppTec credentials and provides SSO for the scheme. AAS is an access authorization service that provides tokens for an authenticated AppTec to access CS and several GW. Therefore, GW is accessed securely, mitigating attacks possibilities, because if an entity is not authenticated and authorized it cannot access the GW. Entities must be previously registered in AAS to reach GW.

We use two key-based encryption level to provide end-to-end secure communication between CS and App. In the first level, CS and App use a secret-key encryption, based on a symmetric master key (KKM), to distribute the session key (KEK). We also use KEK as KD due to its short usage lifetime. In the second level, CS, AppTec, GW, AS and AAS use public-key encryption, and GW and App use secret-key encryption to protect the transmitted data, including KEK, per message. CS and App share KKM that is manually stored into CS system and App firmware, following the ANSI X.9.17 standardization. CS stores the KKMs linked to its App correspondent serial numbers, made in the appliances production process, as said before. Messages exchanged between CS and App are encrypted by KKM, not allowing any intermediate access to the message content, in this case, transporting the session key (KEK). The same protection is obtained after, when KEK encrypts the message data. We assume that KKM is immutable; however, it could be easily updated in the CS and App, if need, without affecting the proposed scheme and technician work.

Our proposal presents substantial protection for an IdM in the Internet context by using asymmetric keys, protecting the messages exchanged between CS, AppTec, and GW. In the IoT context, we use a symmetric key to encrypt message data because it is appropriate to the limited resources of appliances, although it provides an additional per-message protection, using DTLS.

B. Message Flow

We consider two possible end-to-end communications between App and CS, one started by App and responded by a technician, and another started by AppTec and responded by an App. The communication initiated by App implies a service request to CS, e.g. an appliance activation or a maintenance task request. The communication initiated by AppTec implies a service requested previously by an App or a required intervention, e.g. to update an App firmware.

Figure 2 shows a sequence diagram of an end-to-end communication initiated by App. In general, the messages sequence is the same for any request. A Subject requests to the App a service provided by CS (event 1), supplying a request content (requestValue). App generates and uses a KEK to encrypt the content of messages exchanged during the request lifetime. App also uses KKM to encrypt such KEK and a nonce (e.g. a current timestamp). Then, App sends the encrypted value (encryptedValue) to the GW together with its serial number (serialNumber) and the CS address (costumerURL). GW translates the IoT message to the Internet and forwards the message to the CS (event 1.1.1).

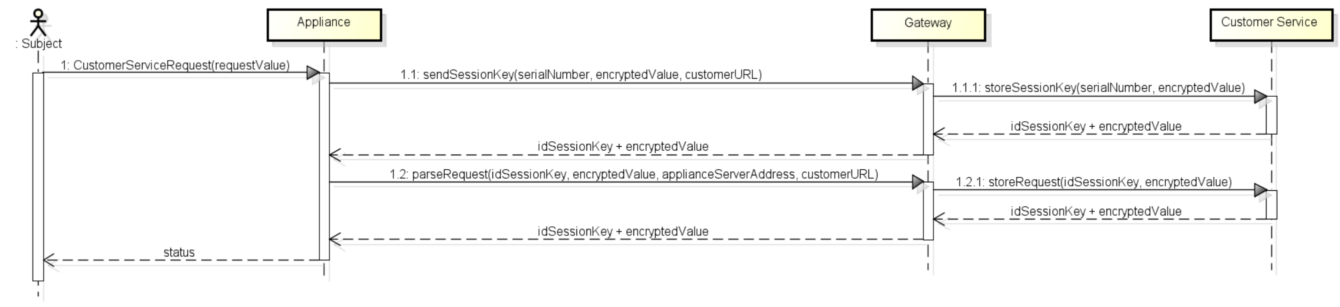
: Subject
Appliance
Gateway
Customer Service
1: CustomerServiceRequest(requestValue)
1.1: sendSessionKey(serialNumber, encryptedValue, customerURL)
1.1.1: storeSessionKey(serialNumber, encryptedValue)
idSessionKey + encryptedValue
idSessionKey + encryptedValue
1.2: parseRequest(idSessionKey, encryptedValue, applianceServerAddress, customerURL)
1.2.1: storeRequest(idSessionKey, encryptedValue)
idSessionKey + encryptedValue
idSessionKey + encryptedValue
status

Figure 2. Message flow for a request initiated by an Appliance.

CS retrieves the App KKM from the serial number present in the message and decrypts the data (encryptedValue). CS validates the App KEK, using the nonce to avoid replay attacks. CS stores the App KEK and uses it to encrypt and decrypt future App messages for the same session. CS replies to App the session key index (a numeric value that identifies the stored KEK) and a replay nonce (nonce + 1). The nonce is used to guarantee the CS authenticity, ensuring that only KKM holder can decrypt and re-encrypt a message containing the replay nonce. GW receives the reply and maps the KEK index with the App address. Then, GW parses the Internet message to an IoT message and forwards the message to the App.

App receives a message encrypted with KEK, decrypts the message, and validates reply nonce, through its value. App encrypts the request with a new nonce and forwards to the GW (event 1.2), supplying the session key index that will be used to communicate to CS. CS retrieves the session key based on the session key index, decrypts and stores the request, to be replied asynchronously by a technician. CS returns the session key index and the reply nonce. App receives the response, validates the reply nonce and informs the Subject a status of the request.

Figure 3 shows a sequence diagram that represents the authentication and access authorization process for a technician to access the AppTec. A technician requests access to AppTec (event 1) and is redirected to AS with its requested credentials (event 2). AS validates the technician credentials and replies a code (with a short valid time) to be used to request an access token to AAS (event 3.1). AAS returns a token to be used by the Technician to respond an App request.

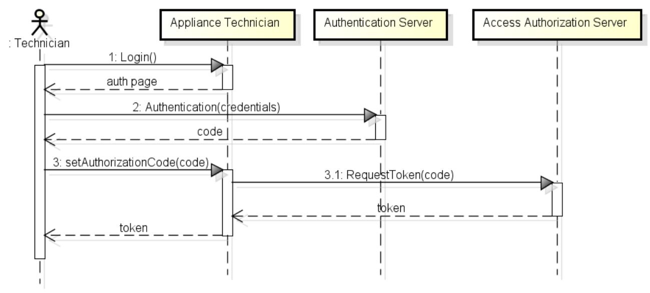

Figure 3. Authentication and access authorization for a technician.

After being authenticated and authorized, the technician has access, using AppTec, to the data request (sent previously by an App). AppTec retrieves KEK and its index, decrypts the request, processes it and forwards the encrypted response to the GW. GW receives the end-to-end encrypted response along with an access token, which after being validated, allows GW to parse the message and forward the encrypted response to the App. App receives the encrypted response and get KEK based on its index, decrypts the response, and processes it.

The second type of communication, initiated by a technician, is applicable when she/he wants to collect information or perform a maintenance task, e.g. an App firmware upgrade. This communication follows the Call Back procedure (Figure 4). Assuming an authenticated and authorized technician, as shown in Figure 3, she/he uses AppTec to request to CS some App data (event 1.0), supplying a token and a serial number (serialNumber). CS validates the token (event 1.1) and answers App data, including the GW address (gatewayAddress) associated to the App. AppTec requests to GW of an App to start a communication with it (event 2.0). GW validates the access token (event 2.1) and notifies the App (event 2.2). App will start a session following the communication steps mentioned in Figure 3.

V. Prototype

In this section, we present a prototype that implements the authentication scheme based on IdM and the proposed access authorization. The prototype uses IT standard, well-known technologies, and open-source coding libraries.

A. Implementation

The Manufacturer Domain consists of two components, AppTec and CS. AppTec was implemented using the framework Vaadin [19], taking advantage of a rich and interactive user experience besides support for a smartphone. CS was implemented as a RESTful web service using the JAX-RS API [20].

The Customer Domain consists of one GW and several Apps. The GW interfaces with Internet context were deployed on an HTTP server implemented in Java. IoT interfaces were deployed through a CoAP server implemented using the Californium library [21]. Thus, GW can parse messages between HTTP and CoAP protocols and vice-versa.

The App was implemented in Java, also based on Californium project, and can be executed on ContikiOS (Operating System for IoT) [22]. We use AES 128-bit algorithm to perform encryption used for KKM and KEK. Scandium, a subproject of Californium, is used to make communication between App and GW since it supports DTLS version 1.2.

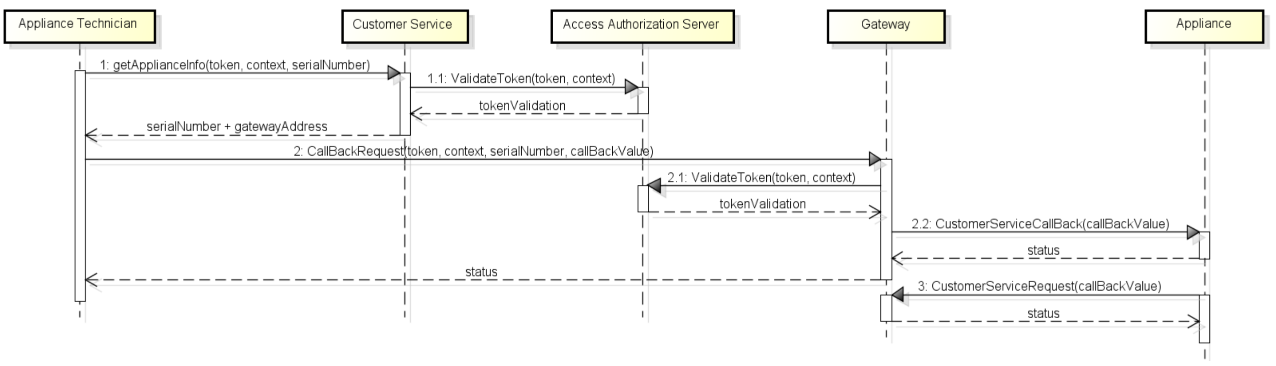
Appliance Technician
Customer Service
Access Authorization Server
Gateway
Appliance
1: getApplianceInfo(token, context, serialNumber)
1.1: ValidateToken(token, context)
tokenValidation
serialNumber + gatewayAddress
2: CallBackRequest(token, context, serialNumber, callBackValue)
2.1: ValidateToken(token, context)
tokenValidation
2.2: CustomerServiceCallBack(callBackValue)
status
status
3: CustomerServiceRequest(callBackValue)
status

Figura 4. Sequence diagram for Call Back procedure.

The Authentication Server (AS) was implemented following the OpenID Connect specification, using the Nimbus library [23] – a Java library that implements OpenID Connect and the OAuth 2.0 specification. Nimbus provides IdM for AppTec, CS, and GW, and it ensures that only authenticated and authorized users access Apps. The Access Authorization Server (AAS) was implemented following the OAuth 2.0 specification and Nimbus, in order to issue access tokens for an authenticated AppTec to access the CS and several GWs.

In the network level, we assume that IPv6 address do not change but could be updated dynamically, without affecting the proposed scheme. In the IoT context, we used software control to reduce the bandwidth allowing the use of 6LoWPAN on IEEE 802.15.4.

Figure 5 shows the prototype architecture highlighting secure communication between components and the used protocol stack. From the Internet viewpoint, communication is made using HTTPS and from IoT using CoAPs.

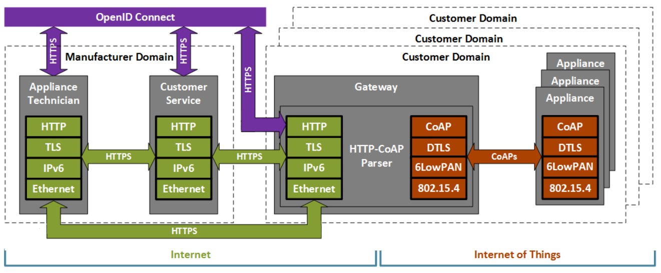

Figure 5. Prototype architecture.

B. *Evaluation*

The evaluation was performed using two machines in a local network to get a controlled environment and to avoid interfering with time measurements. One machine hosts OpenID Connect server, CS, AppTec and GW, and another machine hosts Apps instances. Each machine, Intel i7, has eight cores, 16 GB memory, 1 TB disk, and ran Ubuntu 14.04.2 LTS, Java 1.7.0_75 64 bits. In the Apps machine, we reduced the bandwidth to 40 Kbps in the frequency of 915 MHz, in order to mimic more accurately an App communication, following the 6LoWPAN protocol specification.

The tests are intended to measure the impact of the end-to-end authentication scheme taking into account three issues: (i) the impact of the request size in the response time; (ii) the impact of the Apps number in the response time; and (iii) the impact of the CoAPs in the communication between GW and Apps.

In the tests, we used message sizes vary from 32 bytes to 4096 bytes and the appliances number from 10 to 50, which represents a feasible value in the Smart House context using CoAP and CoAPs.

Figure 6 shows that, for the request size from 32 to 1024 bytes using CoAPs, the response time stays below 1500 ms per request. This observation indicates the proposal works very well, with a nearly constant overhead, even for a large number of 30 or 50 Apps. However, we noted an increased overhead to be raised for messages that have over 1024 bytes, reaching a response time of 1600 ms, when using messages of 2048 bytes. Additionally, messages of 4096 bytes have a response time approximately of 2000 ms per request, but still adequate for the Apps number.

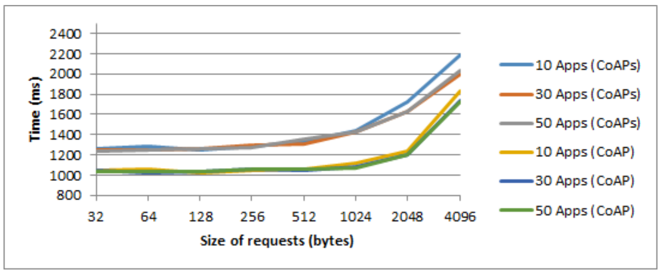

Figure 6. Prototype Evaluation

We also could observe that using CoAP or CoAPs with message size greater than 1024 bytes, the response time increased more for 10 Apps than for 30 or 50 Apps. This result suggests that SSO provides some time advantage when the number of Apps increases over than 10.

We conclude that GW has a good performance when considering a realistic number of App. Furthermore, the results show the seamless integration between the two contexts, without significant impact to IoT. Meaning, there is no significant difference between the response time from 10 to 50 Apps, about 7% to CoAPs and 6% to CoAP.

Considering the size of requests, we can observe that there was an overhead for messages greater than 1024 bytes. However, since it was used symmetric key, suitable for IoT, COAP or COAPs, the proposal did not affect importantly the response time, showing its feasibility in real world applications.

VI. Conclusion

We presented an authentication method integrating IdM from Internet context to IoT for provide it with SSO. The link between the two contexts is provided by a gateway that cannot access message contents, but acts parsing the context from IoT and Internet, and vice-versa.

We were concerned about a single point of compromising from the security perspective, the gateway. Therefore, we used symmetric keys, suitable for IoT to protect messages end-to-end, from appliances to Customer Service. The Technician authentication in the gateway aims to mitigate the possible attacks coming from Internet. Thus, the gateway provides an appliance isolation from Internet, preventing attacks to the IoT devices, less powerful in terms of resources to protect themselves.

The way of the proposal provided SSO, we can say "encapsulated in IdM", is suitable to IoT, without requiring from the IoT appliances an extra effort to interact with an Internet server, as proposed in the literature. Furthermore, technician can access multiple Appliances from a single authentication, not requesting to know a different password for each appliance nor to use the same password for all appliances – practice that submits the appliances to risks, if password is discovered.

The additional protection provided by per-message mechanism improves the method qualitatively while it adapts to each context. Moreover, it makes hardly the content violation since an end-to-end asymmetric key also protects the most sensible parts of the message content.

The proposed scheme was based on IT standards and with a prototype that uses consolidated technologies for Internet and IoT context.

We show the feasibility of our approach analyzing its response time, varying the number of appliances and the size of messages. The proposed approach presented no significant overhead for response time from 10 to 50 appliances and from 32 to 1024 bytes per message. Moreover, the overall response time stays below 1500 ms per request, an acceptable overhead, taking into account it is been provided a key-based IdM for end-to-end security in IoT.

As future work, we will test the proposal with ContikiOS and consider other measures, such as energy consumption.

ACKNOWLEDGMENTS

This work was partially sponsored by the Brazilian National Council for Scientific and Technological Development (CNPq), grants 310671/2012-4 and 404963/2013-7. João Marynowski wishes to thanks the Coordination for the Improvement of Higher Level Personnel (CAPES) for the scholarship granting.

References

[1] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," Comput. Networks, vol. 54, no. 15, 2010, pp. 2787–2805,

[2] S. Babar, P. Mahalle, A. Stango, N. Prasad, and R. Prasad, "Proposed security model and threat taxonomy for the Internet of Things (IoT)," in Proc. of the CCIS - Communications in Computer and Information Science, 2010, pp. 420–429.

[3] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of things: Vision, applications and research challenges," Ad Hoc Networks, 2012, pp. 1497–1516.

[4] A. Belapurkar, A. Chakrabarti, H. Ponnapalli, N. Varadarajan, S. Padmanabhuni, and S. Sundarrajan, "Distributed Systems Security: Issues, Processes and Solutions", John Wiley & Sons, 2009.

[5] N. Sakimura, J. Bradley, M. Jones, B. de Medeiros, and C. Mortimore, "OpenID Connect Core 1.0,". [Online]. Available: http://openid.net/specs/openid-connect-core-1_0.html.

[6] N. Kushalnagar, G. Montenegro, and C. Schumacher, "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals," RFC4919.

[7] Z. Shelby, K. Hartke, and C. Bormann, The Constrained Application Protocol (CoAP), IETF RFC 7252.

[8] E. Rescorla and N. Modadugu, "Datagram Transport Layer Security Version 1.2.", IETF RFC 6347.

[9] ANSI, "X9 Encryption Collection". [Online]. Available: http://webstore.ansi.org/RecordDetail.aspx?sku=X9+Encryption+Collection

[10] J. Liu, Y. Xiao, and C. L. P. Chen, "Authentication and Access Control in the Internet of Things," in Proc. of the ICDCSW - Intl. Conf. on Distributed Computing Systems Workshops, 2012, pp. 588–592.

[11] D. Van Thuan, P. Butkus, and D. Van Thanh, "A user centric identity management for Internet of things," in Proc. of the ICITCS - IT Convergence and Security, 2014, pp. 1–4.

[12] P. Fremantle, B. Aziz, J. Kopecky, and P. Scott, "Federated Identity and Access Management for the Internet of Things," in Proc. of the Int. Workshop on Secure Internet of Things, 2014, pp. 10–17.

[13] M. Leo, F. Battisti, M. Carli, and A. Neri, "A federated architecture approach for Internet of Things security," in Proc. of the EMTC - Euro Med Telco, 2014, pp. 1–5.

[14] S. Cirani, M. Picone, P. Gonizzi, L. Veltri, G. Ferrari, and S. Member, "IoT-OAS: An OAuth-Based Authorization Service Architecture for Secure Services in IoT Scenarios," IEEE Sens. J., vol. 15, no. 2, pp. 1224–1234, 2015.

[15] C. Chibelushi, A. Eardley, and A. Arabo, "Identity Management in the Internet of Things : the Role of MANETs for Healthcare Applications," Comput. Sci. Inf. Technol., vol. 1, no. 2, pp. 73–81, 2013.

[16] R. Hummen, J. H. Ziegeldorf, H. Shafagh, S. Raza, and K. Wehrle, "Towards viable certificate-based authentication for the internet of things," in Proc. of the HotWiSec - Hot topics on Wireless network Security and privacy, 2013, p. 37.

[17] N. Li, Q. Wang, and Z. Deng, "Authentication framework of IIEDNS based on LDAP & Kerberos," in Proc. of the IC-BNMT - Int. Conf. on Broadband Network and Multimedia Technology, 2010, pp. 695–699.

[18] X. Yao, X. Han, X. Du, and X. Zhou, "A lightweight multicast authentication mechanism for small scale IoT applications," IEEE Sens. J., vol. 13, no. 10, 2013, pp. 3693–3701.

[19] Vaadin, "OpenID Integration." [Online]. Available: https://vaadin.com/directory#!addon/openid-integration.

[20] "Java API for RESTful Services." [Online]. Available: https://jax-rs-spec.java.net/.

[21] Eclipse Foundation, "Californium." [Online]. Available: https://www.eclipse.org/californium/.

[22] "The Contiki Operating System." [Online]. Available: http://contiki-os.org/.

[23] "Nimbus OAuth 2.0 SDK with OpenID Connect extensions." [Online]. Available: http://connect2id.com/products/nimbus-oauth-openid-connect-sdk.