Cluster Comput (2017) 20:651–660
DOI 10.1007/s10586-017-0729-3

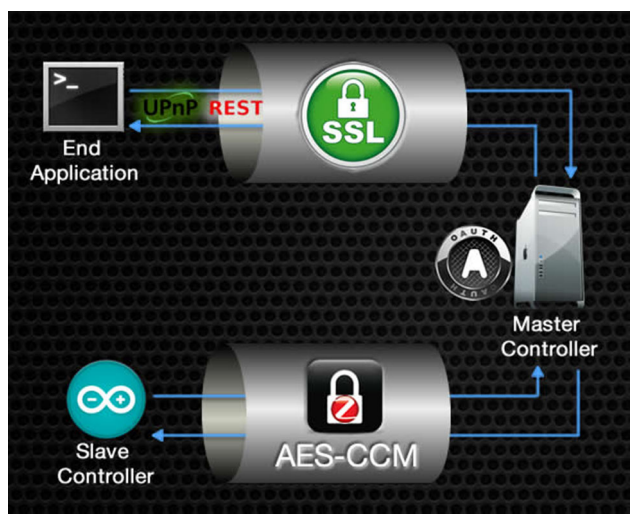
Security analysis of a proposed internet of things middleware

Hiro Gabriel Cerqueira Ferreira[1] · **Rafael Timoteo de Sousa Junior**[1]

Received: 7 November 2014 / Revised: 1 October 2016 / Accepted: 4 January 2017 / Published online: 28 January 2017
© Springer Science+Business Media New York 2017

Abstract This paper proposes security measures for a defined uniform and transparent internet of things middleware, named UIoT. The proposed architecture is deployable and comprises protection measures based on existent technologies for internet security, as well as support for peculiar security needs of the internet of things (IoT). The aim is to provide privacy, authenticity, integrity and confidentiality on data exchange among participant entities in a given IoT scenario yet allowing resource constrained nodes to be part of the network. The main contributions of this work include a brief survey on IoT attack models and possible defenses and proposal of a security model for UIoT, which can be generalized as security measures for other IoT middleware and gateways.

Graphical Abstract



Keywords Internet of Things · Middleware · Security model · Attack model · IoT gateway

1 Introduction

Computing has evolved from the mainframe through the personal computing eras and has been advancing into the ubiquitous Computing paradigm [1]. Miniaturization and reduction of power consumption allowed the integration of processors into common objects and heterogeneous devices, also named things. Wireless communications and the Internet enabled such things to exchange data, send and receive commands and, with the help of sensors and actuators, to be aware of their own state and perform specific duties.

✉ Hiro Gabriel Cerqueira Ferreira
hiro.ferreira@gmail.com
http://uiot.org/

Rafael Timoteo de Sousa Junior
desousa@unb.br
http://uiot.org/

[1] Electrical Engineering Department, University of Brasilia
UnB, Campus Darcy Ribeiro Asa Norte, Brasília,
DF 70910-900, Brazil

These communicating things are commonly named smart objects and the expression Internet of Things (IoT) refers to the computing paradigm that organizes data exchange among these objects and other distributed computing entities so as to provide services to end-users [2,3].

To become fully applied, this paradigm requires a standardization effort regarding issues such as: hardware and software shells to convert plain objects into smart ones, object identification, communication channels between smart objects, and a common communication language between smart objects. The expected standards must respond to requirements regarding network scalability, mobility of nodes, diversity of things, interoperability and overall security. Also, since IoT presupposes the Internet as its main data exchange infrastructure [4], issues related to the Internet architecture are also a challenge, including the unique identification for communicating, and possibly moving, objects, applications and users.

Nevertheless, a common standard architecture is still lacking for IoT, an issue that motivates research in this domain. IoT architectures and middleware depend on a better understanding of IoT requirements and the agreement on technical choices for their real accomplishment. Among these research themes, information security is of paramount interest, particularly regarding authenticity, integrity and confidentiality of communications, as well as the preservation of user privacy in IoT.

In this context, for the sake of accuracy, we should refer not to a single, homogeneous and cohesive Internet of Things, but to the various incarnations or instances of the Internet of Things. Figure 1 illustrates a common situation of a multifunctional mobile device carried by a person who drives a vehicle between home and work. Such a device participates simultaneously in several instances of IoT, each instance being actually an *overlay network* on the fixed and mobile Internet infrastructure. The device has different roles in each IoT instance and the situation continuously changes depending on various factors, in particular the device mobility.

Thus, in addition to those IoT instances involving applications with static devices, possibly with limited and fixed capacities regarding storage, processing and communications, it is important to consider existing and forecast scenarios with smart mobile devices (eg. multifunction cell phones, personal accessories, board controllers vehicles and embedded devices), each of which simultaneously present in several instances of IoT. Such instances can involve potentially different hardware platforms and operating system, IoT middleware, communication protocols, data models, types of applications, security policies, etc.

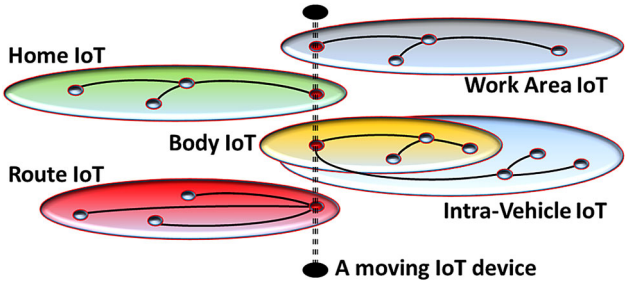


Fig. 1 Different IoT instances for a moving device

As a result, a given IoT device may simultaneously have several roles, from acting as a single sensor under the control of another entity, or operating as an idempotent peer of another device, to being the main controller of a complete IoT application. In this situation, since various IoT services can simultaneously be offered, under different conditions that can change at any time (for example, due to mobility, or the remaining power capacity, or user decisions, etc.), it is important for the device, as well as for the possible applications in which it is contributing, to have sensitivity to the changing context and related security requirements.

Indeed, in accordance to [5], from a conceptual point of view, the Internet of Things concept refers to entities that act as suppliers and/or consumers of data related to the physical world, i.e., the IoT focuses on data and information, instead of communication technologies and protocols. If, on the one hand, the IoT is understood as a highly dynamic system and distributed network, where a very large number of smart objects produce and consume information, on the other hand, there is a prospect whose main issue is to share and integrate functionalities and features provided by smart objects, composing them in the form of services and corresponding generated data streams, both needing appropriate security measures. In this paper, the proposed security model includes protection for information flows and authentication of service providers and consumers.

According to [6], dealing with heterogeneity and scalability is critical in a complex and dynamic system like this, in which the devices, with little human intervention, shall offer self-capabilities, notably high degree of configuration autonomy , self-organization and self-adaptation to various scenarios, self-reaction to events and stimuli, and self-processing of huge amounts of exchanged data. In this paper, we propose to assign responsibilities of that order mandatory to the IoT middleware, but also allowing the device to optionally offering security services to participate in fully distributed security measures, as for example distributed intrusion and anomaly detection and distributed denial of services avoidance, which are required for the operation of efficient services in dynamic environments composed of large number of nodes that can move, causing sudden appearances and disappearances of services at any time.

It is worth to point that these fully distributed security measures correspond to the concept of a cognitive IoT [7], where physical or virtual things are interconnected and inter-

act according to a context sensitive perception-action cycle. This paper argues for the possible registration of security services by initiative of the device, as implemented in the existing IoT middleware [8] that serves for representing the proposed IoT security model.

On the other hand, the IoT can be seen from two perspectives, i.e., one being network or middleware centric while the other is centered on the thing (object or device), the first focusing on services provided by the network using data provided by the devices and the second, focusing on smart device services [9]. In this paper, we combine security services internal to the IoT middleware with distributed security services integrated by the devices when these devices are able to give their contribution to the service.

The paper [10] states that context sensitive applications are able to continually adapt to complex and dynamic environments and functions, in the presence of motion and heterogeneous systems and devices, which requires processes and content regarding two forms of context, physical and virtual. The physical contexts involve perceptions of sensor devices, such as speed, acceleration, pressure, light, location, movement, sound, touch and temperature. Virtual contexts are those specified by users or captured from user interactions, preferences, interests, goals and tasks. In this paper, we consider that IoT security arises from the integration of the two views, offering, in the form of services, operational capacity and proper information for securing each IoT application instance. In this paper proposition, the security model is applied within a context life cycle [11] used by the IoT application or instance.

Also, since security is a prerequisite for each IoT instance, this paper holds the argument of [12] according to whom the application operating logic should be separated from its security enforcement policy. So, this paper is dedicated to a functional proposal for an IoT security model that can generically be used by any IoT application.

In view of these considerations, based on a previously proposed middleware, named UIoT, and its usage scenarios, respectively presented in [13] and [14], this paper presents and discuss attack and security models for IoT. These models consider envisioned environments where devices are able to automatically provide services to users, i.e., services that must work without human intervention [1,2].

Also, the security model proposed in this paper is aimed at reusing existing widespread security technologies and must respond to scalability requirements to cope with an expected significant usage growth rate [9], while preserving users privacy [15]. Moreover, this security model must consider the limitations of smart objects and devices regarding their energy and computational resources, which constrain security logic and algorithms.

The remainder of this paper is organized as follows. Section 2 briefly discusses middleware and security technologies applicable to IoT. Section 3 presents vulnerabilities and attack models for IoT. Section 4 describes our proposed security model. Section 5 concludes this paper pointing some correlated open research topics.

2 Middleware and security for IoT: related work

This section presents a brief survey of IoT middleware proposals, as well as existing Internet security technologies that can be used in IoT scenarios. This type of middleware aims to allow things to communicate with internal and external entities while these security technologies integrate measures to provide privacy, authenticity, integrity and confidentiality to such communications. Thus, the technologies presented in the following subsections are correlated to the proposed security model described in Sect. 4.

2.1 Related IoT middleware and architectures

Since the middleware and architectures presented hereafter are intended to enable compliant objects to communicate with other entities, they should bring interoperability and security to these devices and entities. Their components should be deployable, transparent [1], scalable and should be responsible for most processing and storage, given that the mentioned middleware and architectures must enable heterogeneous devices to be part of IoT environments even if the devices are severely constrained on processing, storage and energy capabilities.

Since the research related to this paper intends to use existing solutions to serve as base for its experimentation and propositions, existing middleware, compliant to the requirements in the preceding paragraph, were analyzed to produce a brief overview that is presented hereafter on relevant initiatives.

2.1.1 LinkSmart

The Hydra middleware, lately renamed LinkSmart [16], was first designed to enable networking for embedded systems. It has evolved to a IoT like philosophy and now provides a deployable service-oriented architecture with embedded security.

To configure an IoT instance, this middleware requires the configuration of LinkSmart devices, an operation that is performed by loading hardcoded scripts on LinkSmarts main software (proxies) to adapt devices to LinkSmart's open source protocols. Such middleware demands a relatively important amount of processing capacity from devices, what makes it not simple to deploy for most legacy plain objects. For these reasons this middleware was not selected for this paper proposal.

2.1.2 IoT-A

IoT-A [17] is a research collaboration funded by IETF under the ICT Theme of Framework Programme 7 (FP7). The research team delivered a book and an IoT architectural reference model (ARM).

Unfortunately, such research does not yet provide a prototype, even-though it proposes a complete ARM for IoT, to be used for measuring data, which allows improvements and validations, or to be adapted to other usage scenarios. By far, it is not a simple task to deploy a fully compliant middleware to such ARM, though it constitutes a very complete reference. Since IoT-A lacks a prototype it was discarded from this work consideration.

2.1.3 iCore

The iCore Project [18], also funded by IETF under the ICT Theme of Framework Programme 7 (FP7), lies in its requirements gathering phase for future development. It has some implementations and promises to be scalable, embracing a device discovery service, an eventing service and encompassing heterogeneity of devices.

The iCore middleware is related to IoT-A and aims to empower IoT through cognitive technologies. It is based on the concepts of virtual objects (VOs), composite virtual objects (CVOs) and functional blocks. VOs are logical abstractions of heterogeneous devices prone to interoperability. CVOs are service providers composed by a group of cognitive VOs. Functional blocks exists to represent other IoT stakeholders. Since this proposal is a work in progress and has no deployable deliverable, it was not chosen for this work.

2.1.4 UIoT

UIoT was first proposed in [13] as a communication architecture to embrace both legacy and new developed IoT devices. The corresponding use cases were proposed in [14]. UIoT architectural model, represented in Fig. 2, provides uniform and transparent APIs to applications as well as subscription services. This middleware concentrates administrative tasks and provides mobile serial digital and analog interfaces to communicate with end devices. UIoT abstracts devices complexity by means of the UPnP device model [19], which is manageable by administrators through web forms and REST APIs. Despite it was developed independently of other proposed architectures, UIoT presents many similarities to them, but focusing on the uniformity of services and the reduction of load for end devices. Also, UIoT shares the same principles of 6LowPAN [20].

Regarding operations to convert plain objects into smart objects or to aggregate smart objects under its uniform APIs,

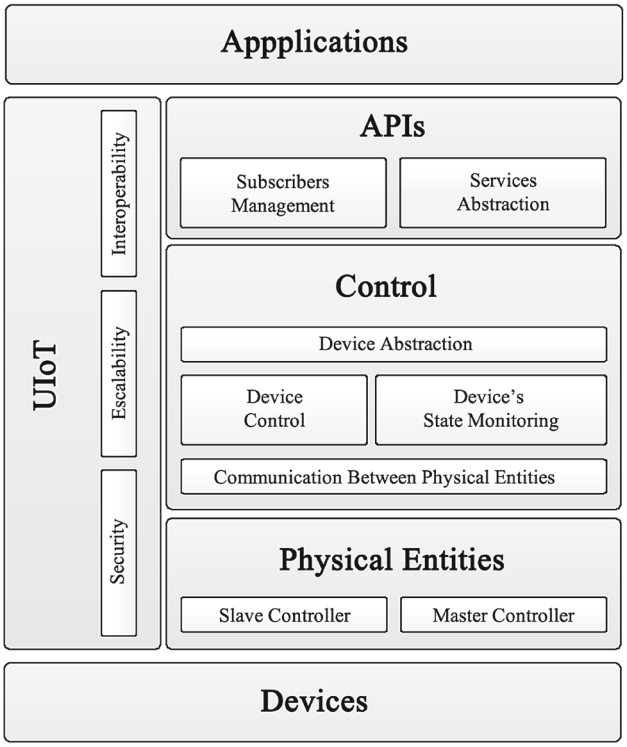


Fig. 2 UIoT architectural model

UIoT performs two routines named control routine and state monitoring routine. The control routine consists in executing requests on devices in order to change their state to achieve what the applications desire, hiding operation complexity from them. The state monitoring routine is intended to keep track of devices state in order to notify state transitions to state subscribers (eventing).

UIoT internal structure comprises two types of entities: the master controller and slave controllers. While the master controller is responsible for keeping state and abstractions and communicating with applications, the slave controllers are responsible for communicating with devices through their digital and analog interfaces. To promote scalability, these entities can be configured and decoupled in various computing platforms, raging from the Raspberry Pi platform [21] to Arduino boards [22] or computer grids in clouds, allowing the adaptation to the application scalability scenario. Specifically, the master controller must be able to abstract devices, hold services, data and convert applications requests to simple commands like "read pin value" or "write value to pin" and send them to slave controllers. Slave controllers must be able to write values to its pins and read values from them. Communication between such entities is made through the ZigBee mesh protocol [23].

UIoT's layered architecture allows new APIs to be added to the Service Layer, as well as other communication models, like PowerLine or bluetooth or WiFi, to the Communication

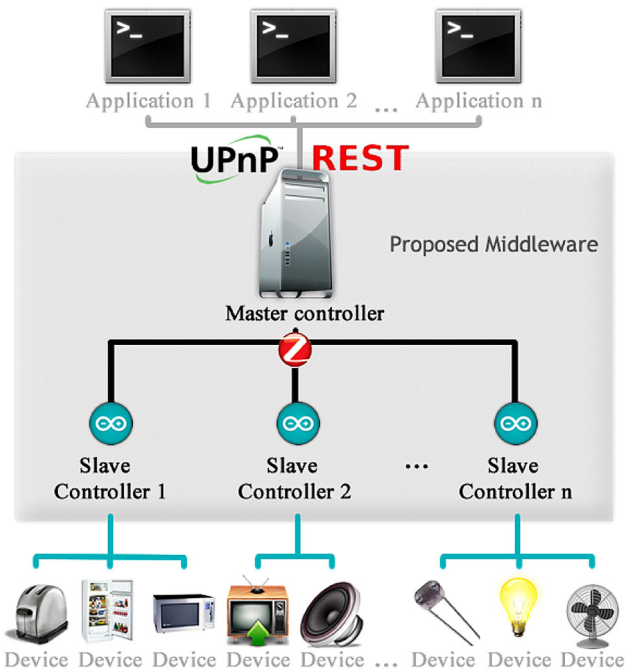


Fig. 3 Components of the UIoT prototype

layer while newly developed devices can be added to the execution layer. If this proposed transparent communication model is kept, it can be extensively extended to meet any scenario.

The first prototype of UIoT, represented in Fig. 3, was built with a set of existing technologies in order to make it understandable and extensible by other research efforts. Thus, in this prototype, REST [24] and UPnP [19] APIs are provided for applications while the ZigBee mesh is used for communication among Master and Slave controllers. The usual TCP/IP suite is used for communications involving the Master Controller and the applications. An Arduino [22] with XBEE radio expansions implements the slave controllers and parses master controller instructions to end devices. This prototype was designed to cope with IoT scalability, as well as the diversity and the computational and energy limitations of devices.

Since it has an operational prototype and its architecture responds to general requirements common to other IoT middleware, UIoT was considered appropriate to be used in this paper study without any loss of generality. Besides, since its security model was yet to be proposed, it was chosen as the base middleware for this work.

2.2 Related security technologies

Given the characteristics of IoT middleware and the utilization of the Internet for the interoperability among IoT entities, there are security technologies commonly used in the Internet that can be leveraged for IoT implementation. The technologies briefly presented hereafter are commonly used within distributed scenarios such as the World Wide Web, wireless sensors and actuators networks (WSAN), and Mobile Ad Hoc Networks (MANETs), which are considered as precursors and enablers of IoT.

2.2.1 AES-CCM

AES-CCM is an operation mode of the Advanced Encryption Standard (AES). It is designed to provide privacy and authenticity in constrained environments and compact implementations [25], using counter mode added to cipher block chaining with message authentication code (CBC-MAC) [26]. Most WSANs use ZigBee [23], commonly implemented in IEEE 802.15.4 chips which set AES-CCM by default. As a result of its vast usage to secure WSANs while keeping scalability [23], ZigBee with AES-CCM is suitable for IoT scenarios.

2.2.2 TLS

Transport Layer Security (TLS) is a connection-oriented protocol designed to secure data exchange between applications [27]. It uses symmetric (usually, AES) and asymmetric (usually, RSA) cryptography schemes to ensure privacy, authenticity and integrity in a communication session between client and server. An application protocol stack with HTTP over TLS, also known as HTTPS [28], is largely used by Internet services because it allows secure data exchange between clients and servers, such as web browsers and web servers.

Considering the Internet as the natural infrastructure for data exchange between distributed IoT devices, TLS seems to be a fair choice to secure their communications sessions [29]. This way, it is also possible to keep interoperability throughout the World Wide Web.

2.2.3 oAuth

The designation oAuth 2.0 refers to a low complexity authentication protocol that allows an entity to access resources belonging to other entities. This access authorization has a limited scope, defined by the resource owner by means of an access token [30]. This authentication and authorization suite is used by large enterprises (Google, Facebook, Windows live, GitHub and others) demonstrating its scalability to authenticate multiple devices and applications. In the case of such enterprises, oAuth generally manages applications access rights to users' resources [31] so that these resources are only accessible to the extent their owner users allow, for the right amount of time, without requiring the users to give their passwords to applications.

IoT environments might require users and applications to authenticate themselves before accessing devices' resources. In such cases every entity should be uniquely identifiable by the devices' manager before retrieving data from, or invoking actions on, the concerned devices. That way, the manager will be able to distinguish each requester and give appropriate access to devices' resources. Since IoT demands a simple, robust and secure authentication method and common sense tells that credentials like passwords should only be known by their owners, oAuth comes as a good candidate to solve authentication and authorization issues within IoT environments.

3 Analyzed attack models and possible defenses

This section is devoted to define attack models against which we intend to protect the IoT middleware and its applications. By analogy to cases described in [32], we have identified the following major vulnerabilities and attack models applicable to IoT: Passive Listening, Impersonation, Black Holes, Sybil Attacks, Denial of Service and Physical Theft, that are each one introduced in the following subsections using UIoT as the reference implementation for the analysis.

3.1 Passive listening

Entities listening to conversations between applications, controllers and devices, without authorization characterize Passive Listening. Such vulnerability can be prejudicial when listeners collect data in order to perform future malicious actions.

Default ZigBee communications and HTTP requests are made in plain text. Thus, any entity capable of listening the TCP/IP network in promiscuous mode can trace interactions among the UIoT master controller and the applications. Also, any entity capable of listening to IEEE 802.15.4 defined radio channels can trace interactions among the master and slave controllers. To be mitigated, this vulnerability requires private conversations among UIoT authentic entities.

3.2 Impersonation

Impersonations comprehend attacker entities pretending to be IoT devices, controllers or applications. Such entities can collect and hold data in order to perform future malicious attacks like black hole and sybil attacks. A common operating mode is the man-in-the-middle attack, in which an entity fakes to be the legitimate server for a client and simultaneously to be the legitimate client for this server, thus gaining access to information from the two attacked parts.

If an attacker entity is capable of communicating using ZigBee or HTTP, it will also be able to stay in the middle of a conversation among other UIoT authentic entities. This attacker will be able to parse messages, having access to all exchanged information, and making the authentic entities to believe there is no other entity in the middle. Overcoming this vulnerability requires to uniquely identify UIoT authentic entities. Thus, an authentication method should run over private channels in order to mitigated Impersonation.

3.3 Black holes

Black holes can be exploited by malicious entities that occupy privileged physical locations and join the controllers mesh network without authorization, faking to being a regular part of such environment. The attacker then convinces its neighbors to be a trustful traffic router, but does not forward the received data, thus behaving as a sink node for such network.

To avoid this sort of attack, malicious entities should not be able to join UIoT networks, meaning that effective authentication methods are required for granting access to legitimate entities.

3.4 Sybil attacks

In sybil attacks, an arbitrary number of malicious entities collude to use UIoT in order to perform unauthorized actions.

By faking a controller, the attackers could join the mesh network without authorization and send misleading control messages to other controllers, thus manipulating devices, or faking responses to service layer requests, thus precluding service provisioning to applications.

Another sybil attack can be made by entities faking to be devices in order to send misleading state notification messages in the name of authentic devices or to send misleading control messages to networked controllers. Using passive listening, an attacker could get session details of applications and send messages in the name of these applications.

In the same way as Black Holes, to counter sybil attacks authentication methods are required, as well as privacy and confidentiality in conversations during all times, including in authentication operations, since a sybil node could get authentication credentials and change the packets to whatever it desires if channels are in plain text.

3.5 Distributed denial of services

In Distributed Denial of Services (DDoS) attacks, malicious entities could perform, from different servers, a possibly large amount of requests, causing master controllers to overload and not respond to any other request.

In order to prevent such attack, a possible counter-measure is that the master controller should count and correlate received massages in order to filter and attend authentic requests while discarding DDoS classified messages. It is

worth to observe that this is not a general measure to counter other variants of DDoS, such as those originating in IoT devices against entities in other IoT instances or in other networks. This seems to require fully distributed countermeasures as described in Sect. 4.4.

4 Proposed security model

As described in Sect. 2.2, AES, oAuth, and TLS are very well defined and accepted security technologies that can be integrated into the UIoT architecture so as to fulfill this architecture security requirements.

Within UIoT, the master controller has no computational nor energy constraints oppositely to end applications and slave controllers which are very heterogeneous entities and may present computational or energy limitations. Most end devices are passive plain objects controlled by slave controllers, and their security is out of our scope, as we aim to protect interactions of smart objects which are abstractions generated by our middleware shelling the plain objects.

This section covers the middleware internal security aspects, and those regarding its interfaces to the external world, as well as authentication procedures and the smart objects access control list (ACL). Internal security comprehends communication between master and slave controllers, while external interfaces security encompasses communication between the master controller and applications or between the master controller and system users.

Authentication and role based ACL are proposed for applications and users interactions with the master controller, which in turn generate commands that will be applied to end devices by slave controllers. The proposed model responds to the need of unique identification for communicating, and possibly moving, objects, applications and users. This is more than a question of IP addresses, but a more general identification question considering mobility and the possible participation of devices and users in different IoT instances. For this reason, we adopt the idea of the middleware attributing the identification to each device and the security model to having the structure and processes for this identification and the corresponding subsequent authentication and authorization, as required in our attack models (Sect. 3) and integrated in the abstraction of devices and users for purposes of authentication and access control list described in Sect. 4.3.

Also this section discusses the possibility of the middleware offering support to fully distributed security measures, as for example distributed intrusion and anomaly detection, distributed denial of services avoidance, distributed trust and reputation, etc.

The proposed security model based on regular technology for UIoT is represented in Fig. 4 whose components are discussed in the following sections. To our best knowledge this is the first comprehensive published model that can be extended for any IoT middleware.

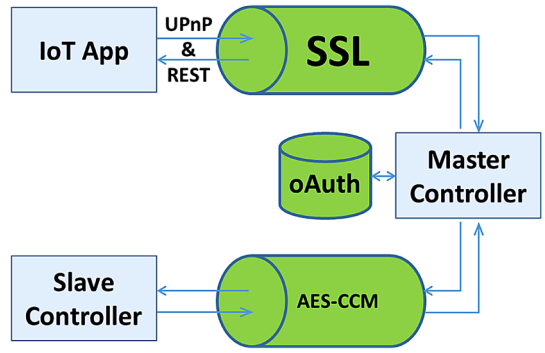


Fig. 4 UIoT core security model

4.1 Internal security

Slave controllers are supposed to be entities possibly with few energy and computational resources so they must not handle too much policies and information processing tasks. This is the reason why this type of entity only deals with cryptography and integrity checks that comprise the ZigBee technology, which is already embedded on slave controllers [13].

ZigBee with AES-CCM using 128 bits key answers this requirement. It allows secure communications by providing confidentiality with block ciphering, integrity with the message integrity code (MIC) and authenticity with the message authentication code (MAC) [25].

Aiming at providing security to the connection between these devices ZigBee entities, it is necessary to configure the ZigBee modules in the master and slave controllers so as to only accept connections with encryption enabled. To make this possible, it is necessary to previously share the network key, and set ZigBee parameters enabled encryption (EE) to 1 and security police handler? (EO) to 0. With this configuration, only nodes that already know the key will be able to join the network of master and slaves controllers, and consequently communicate with end applications.

There is the possibility of performing the automatic initialization of the network keys, for example, using a reliable key distribution and certification entity (certificate authority) or obtaining a validation by a subset of peer entities able to generate a key from a subset of key parts each belonging to a pair entity (threshold cryptography). In both cases, there is an initial process of discovery of trusted entities, which is quite complex and always requires availability of connection with such entities, possibly located outside the IoT network instance. Even so, each device would have to be produced with a certified key. For these reasons, we opted for manual key sharing, by initiative of the owners of devices. In this

way, we avoid both the complexity of the automatic startup and the possible new security loopholes in the automated exchange of keys. Since the device already has to be inserted in the physical environment by an administrator, at that time this administrator can add the key to the device abstraction in the middleware, simplifying the process and initializing the correct key exchange process.

Then, periodically, the master controller sends a new network key to its slave controllers, which are required to persistently save this key for their ZigBee modules. This operation turns the ambient even more secure against eavesdroppers.

4.2 Security of external interfaces

Being defined as the most intelligent entity of the middleware, the master controller centralizes most of the logical procedures and executes most of the hard processing tasks [13]. It is the central middleware processor, the router between applications and slave controllers, as well as the provider of abstractions of end devices.

Since the master controller is the entity that has contact with the external world, it holds the role of securing that channel. The middleware interacts with end applications only through UPnP and REST APIs [13,14], both using HTTP. For that reason, the master controller should perform its unicast communications over HTTPS [28], thus using TLS to enforce security, using AES and RSA to provide confidentiality, and MIC, MAC and SHA-1 to provide data integrity. Prevention against DDoS can be done using one of the algorithms analyzed in [33].

4.3 Authentication and access control list

For authentication purposes, related both to using and managing the entire middleware, four system user types are proposed: root user, plain object owner, smart object user and application user, whose roles are described as follows.

Root users can create all the other types of users and manage all smart objects and their services. This powerful user is supposed to exclusively manage only the master controllers abstractions and, thus, cannot use any of the APIs services. This user can login with a browser at the administration URL http://master_controller_ip/configure [13] and create, view, edit and delete all system users, all system roles, all slave controllers and all end devices with their services and state variables. His credentials are hold by his session.

Plain object owners are created by root users and can create, update, view and remove end devices, system users and user roles. Plain object owners can login at the administration URL and they are supposed to physically add new plain objects at already deployed slave controllers, then add and manage these devices at the master controller. For that reason, one such user can only access entities that he has created.

Smart object users can login at the administration URL and actually use services to which they have allowance, as given by root users and plain object owners.

But, if an Object user wants to use other end applications, before the middleware start exchanging information with end devices, these applications are required to authenticate with the Master controller.

Applications trying to perform actions and receive updates from end devices in the name of smart object users should not know user credentials and must only access services to which the user allows them to. The authentication methods suite proposed by oAuth 2.0 aims to keep the resources and their owner protected. In the context of the UIoT middleware this feature represents the protection for end devices services and smart object users. End applications will be able to get authorization grants without using the resource owner credentials, meaning to the smart object user that his username and password are safe. For that reason, the oAuth method Implicit Authorization Grant [30] is adopted for this case.

To get the authorization grant, end applications have to first redirect the end user to the master controller URL so that a smart object user has to login. If he logs in with success, the master controller will ask him to allow the named application to access the named service. If allowed by the user, an access token is delivered to the application. If the user does not allow access to the service, the requester will be redirected with an empty token. This methodology allows the master controller to keep track of what the application has being doing for further audits, if necessary.

With the specified user types, and the related authentication procedures (the regular login and the two oAuth token methodologies), it is possible for the master controller to limit the access of all middleware resources to only allowed entities. The system users can be controlled separately and each one has its very specific purpose, as discussed in preceding paragraphs. The role based methodology copes with the diversity of end applications, end users and devices, because it simplifies the process to grant authorization (or not) to end users and applications on multiple devices at the same time by only adding or removing a user from a role [5].

4.4 Support to fully distributed security measures

Some IoT security measures must not necessarily be implemented inside the middleware, but just count on the middleware support to be implemented in the form of applications collaborating with devices in fully distributed operations.

Since the devices are able to announce their services, they can propose local security services to be used for the distributed security applications.

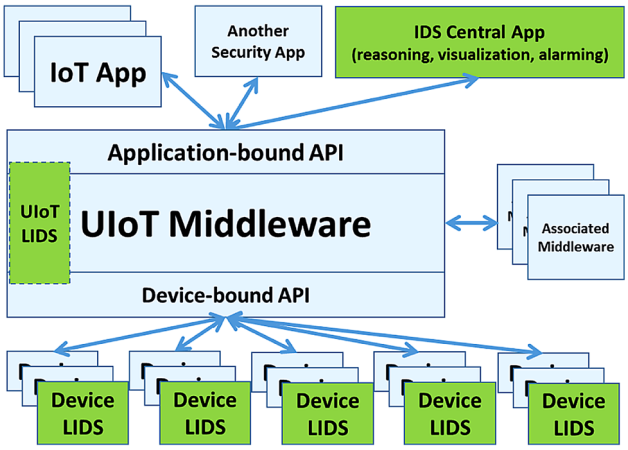


Fig. 5 A fully distributed IDS for IoT

Using the same principles of a fully distributed model proposed for MANETs by [34], IoT instances can count on these fully distributed security measures that can be organized using the devices local security services cooperating with central application hubs. For instance, a smart device can embed a local intrusion detection system (LIDS), a local anomaly detection module, a local autonomic trust reasoning module, a local DDoS probe, a local logbook, and so. These probes can announce their services throughout the IoT middleware to be used by the corresponding distributed IoT security application.

Figure 5 exemplifies the implementation of a distributed intrusion detection system where an IDS reasoning, visualization and alarming central application collaborates with local intrusion detection systems (LIDS), these ones located in smart devices and even within the IoT middleware itself.

The IoT middleware contributes to those fully distributed security measures by protecting the information exchange, the identification of distributed entities and their authorization to interoperate using the middleware internal security services and resources, as well as its external interface.

5 Conclusions

The themes of middleware and security for Internet of Things, as well as implementation issues, are currently the focus of many researchers. The effort of standardization in this domain is yet being developed.

Referencing the proposed transparent and deployable UIoT middleware, with its detailed use scenarios, this paper analyses threats against IoT and specifies the related security architecture, mostly based on existing and robust technologies. The resulting secure middleware allows the implementation of IoT environments, its improvement by other communities and researches, and moves the IoT paradigm closer to reality.

Regarding specific security aspects, this work details how communication and authorization should occur in order to provide privacy, authenticity, integrity and confidentiality to IoT users, applications and devices. The proposed model considers and solves issues on scalability, diversity of devices and applications, simplicity, robustness and keeps possible the interactions with entities of little computational capabilities and limited energy resources.

It is interesting to note that IoT things can embed autonomic countermeasures to respond when they are attacked, similarly to nodes in other types of networks such as MANETs [35]. These measures might include even physically fighting back, an important subject in the area of cyber defense and a subject for further research.

Next steps on UIoT include providing the wrap for devices that already communicate using other technologies. These devices should be put under the master controller realm to get them available throughout the middleware API, bringing uniformity to IoT communications. Technologies such as DTMF, UPnP, Power Line and 6LoWPAN/CoAP allow devices to communicate, but each one using different protocols and grammars. Harmonizing these devices under master controller rules would allow applications to communicate with them always using the same language. Another important issue to be solved is the mobility for end devices and, in that case, how to keep them uniquely identifiable and how to automatically transport their configurations and permissions to other master controllers.

Acknowledgements The authors wish to thank the Brazilian research and innovation Agencies CAPES (Grant FORTE 23038.007604/2014-69), FINEP (Grant RENASIC/PROTO 01.12.0555.00), and the Brazilian Ministry of Planning, Budget and Management (Grant DIPLA 005/2016).

References

1. Weiser, M.: The computer for the 21st century. Sci. Am. **265**(3), 94–104 (1991)
2. Giusto, D., Iera, A., Morabito, G., Atzori, L. (Eds.): The Internet of Things. Springer, New York. ISBN: 978-1-4419-1673-0 (2010)
3. Ashton, K.: That 'internet of things' thing. RFID J. www.rfidjournal.com/article/print/4986 (2009)
4. Weber, R.H.: Internet of things new security and privacy challenges. Comput. Law Secur. Rep. **26**(1), 23–30 (2010)
5. Miorandi, D., Sicari, S., Pellegrini, F., Chlamtac, I.: Internet of things: vision, applications and research challenges. Ad Hoc Netw. **10**(7), 1497–1516 (2012)
6. Borgia, E.: The internet of things vision: key features, applications and open issues. Comput. Commun. **54**, 1–31 (2014)
7. Wu, Q., Ding, G., Xu, Y., Feng, S., Du, Z., Wang, J., Long, K.: Cognitive internet of things: a new paradigm beyond connection. IEEE Int. Things J. **1**(2), 129–143 (2014)

8. Silva, C.C.M., Ferreira, H.G.C., Sousa Jr, R.T., Buiati, F., Villalba, L.J.G.: Design and evaluation of a services interface for the internet of things. Wirel. Pers. Commun. **91**, 1711–1748 (2016)
9. Gubbi, J., Buyya, R., Marusic, S., Palaniswami, M.: Internet of things (IoT): a vision, architectural elements, and future directions. Future Gener. Comput. Syst. **29**(7), 1645–1660 (2013)
10. Zhang, D., Huang, H., Lai, C.-F., Liang, X., Zou, Q., Guo, M.: Survey on context-awareness in ubiquitous media. Multimed. Tools Appl. **67**(1), 179–211 (2013)
11. Perera, C., Zaslavsky, A., Christen, P., Georgakopoulos, D.: Context aware computing for the internet of things: a survey. IEEE Commun. Surv. Tutor. **46**(1), 414–454 (2014)
12. Rizzardi, A., Sicari, S., Miorandi, D., Coen-Porisini, A.: AUPS: an open source AUthenticated publish/subscribe system for the internet of things. Inform. Syst. **62**, 29–41 (2016)
13. Ferreira, H.G.C., Canedo, E.D., Sousa Jr, R.T.: IoT architecture to enable intercommunication through REST API and UPnP using IP, ZigBee and Arduino. In: IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), pp. 53–60 (2013). doi:10.1109/WiMOB.2013.6673340
14. Ferreira, H.G.C., Sousa Jr., R.T., Canedo, E.D.: A ubiquitous communication architecture integrating transparent UPnP and REST APIs. Int. J. Embed. Syst. **6**, 188–197 (2014)
15. Atzori, L., Iera, A., Morabito, G.: The internet of things: a survey. Comput. Netw. **54**, 2787–2805 (2010)
16. Hydra Middleware/Link Smart Middleware. http://www.hydramiddleware.eu. Accessed 18 Feb 2014
17. IoT-A, Internet of Things Architecture. http://www.iot-a.eu. Accessed 18 Feb 2014
18. iCore Project. http://www.iot-icore.eu. Accessed 18 Feb 2014
19. Understanding Universal Plug and Play. http://www.upnp.org/download/UPNP_understandingUPNP.doc. Accessed 18 Feb 2014
20. Shelby, Z., Bormann, C.: 6LoWPAN: The Wireless Embedded Internet. Wiley, New York (2011)
21. Raspberry Pi. http://www.raspberrypi.org/. Accessed 18 Feb 2014
22. Arduino. http://www.arduino.cc/. Accessed 18 Feb 2014
23. ZigBee Alliance. http://www.zigbee.org. Accessed 18 Feb 2014
24. Stirbu, V.: Towards a RESTful plug and play experience in the web of things. In: IEEE International Conference on Semantic Computing, pp. 512–517 (2008). doi:10.1109/ICSC.2008.51
25. McGrew, D., Bailey, D.: RFC-6655 AES-CCM Cipher Suites for Transport Layer Security (TLS). IETF (2012)
26. Whiting, D., Housley, R., Ferguson, N.: RFC-3610 Counter with CBC-MAC (CCM). IETF (2003)
27. Dierks, T., Rescorlar, E.: RFC-5246 The Transport Layer Security (TLS) Protocol Version 1.2. IETF (2008)
28. Rescolar, E.: RFC-2818 HTTP Over TLS. IETF (2000)
29. Iwendi, C.O., Allen, A.R.: Enhanced security technique for wireless sensor network nodes. In: IET Conference on Wireless Sensor Systems (WSS 2012), pp. 1–5 (2012). doi:10.1049/cp.2012.0610
30. Hardt, D.: RFC-6749 oAuth 2.0h Authorization Framework. IETF (2012)
31. oAuth, open standard for authorization. http://oauth.net/2/. Accessed 18 Feb 2014
32. Sousa, Jr., R.T., Puttini, R.S.: Trust management in ad hoc networks. In: Trust Modeling and Management in Digital Environments: From Social Concept to System Development, pp. 224–249. IGI Global (2010). doi:10.4018/978-1-61520-682-7.ch010
33. Mirkovic, J., Reiher, P.: A taxonomy of DDoS attack and DDoS defense mechanisms. ACM SIGCOMM Comput. Commun. Rev. **34**(2), 39–53 (2004)
34. Puttini, R.S., Percher, J-M., Me, L., Sousa, Jr., R.T.: A fully distributed IDS for MANET. In: Proceedings of the Ninth International Symposium on Computers and Communications ISCC, vol. 1, pp. 331–338. IEEE (2004)
35. Adnane, A., Bidan, C., Sousa Jr., R.T.: Trust-based security for the OLSR routing protocol. Comput. Commun. **36**(10), 1159–1171 (2013)

Hiro Gabriel Cerqueira Ferreira graduated in Communication Networks Engineering, from the Universidade de Brasilia, DF, Brazil, in 2012. He received his Masters degree in Electrical Engineering from the Universidade de Brasilia with major domain in telecommunication and specific area in internet of things. He is co-founder of Green Tecnologia, a Brazilian software development company, and is a researcher at the Network Engineering Laboratory, at the University of Brasilia.

Rafael Timoteo de Sousa Junior was born in Campina Grande, PB, Brazil, on June 24, 1961. He graduated in Electrical Engineering, from the Federal University of Paraba UFPB, Campina Grande PB, Brazil, 1984, and got his Doctorate Degree in Telecommunications and Signal Processing, from the University of Rennes 1, Rennes, France, 1988. He worked as a software and network engineer in the private sector from 1989 to 1996. Since 1996, He is a Network Engineering Professor in the Electrical Engineering Department, at the University of Braslia, Brazil. From 2006 to 2007, supported by the Brazilian R&D Agency CNPq, He took a sabbatical year in the Group for the Security of Information Systems and Networks, at Ecole Superiure dElectricit, Rennes, France. He is a member of the Post-Graduate Program on Electrical Engineering (PPGEE) and supervises the Decision Technologies Laboratory (LATITUDE) of the University of Braslia. His field of study is distributed systems and network management and security.