**Cluster Computing: the Journal of Networks, Software Tools and Applications manuscript No.**
(will be inserted by the editor)

# Security Analysis of a Proposed Internet of Things Middleware

**Hiro Gabriel Cerqueira Ferreira** · **Rafael Timoteo de Sousa Junior**

Received: November 7th, 2014 / Accepted: date

**Abstract** This paper proposes security measures for a defined uniform and transparent Internet of Things (IoT) middleware, named UIoT. Focused on bringing real life objects to the virtual realm, the proposed architecture is deployable and comprises protection measures based on existent technologies for security. Protections articulating AES, TLS and oAuth were chosen to provide security services to UIoT parts against described attack models. 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 survey on IoT attack models, proposal of a security model for UIoT, a simplified and generic IoT security module for IoT gateways for things.

**Keywords** Internet of Things · Middleware · Security Model · Attack Model · IoT Gateway

## 1 Introduction

Computing has evolved from the mainframe trhough 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.

These communicating things are commonly named smart objects while 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. Since IoT presupposes the Internet as its main data exchange infrastructure [4], the unique identification for objects is also a problem due to ever-limited availability of IPv4 addresses.

Nevertheless, a common standard architecture is still lacking for IoT, an issue that motivates research. Architectures and middleware must aim at a better understanding of IoT requirements and the agreement on technical choices for its real enablement. Amongst these research themes a relevant one regards the issues of information security, particularly authenticity, integrity and confidentiality of communications, as well as the preservation of user privacy.

Based on a previously proposed middleware, named UIoT, and usage scenarios, respectively presented in [5] and [6], this paper presents and discuss attack and security models for IoT using widespread security technology. These models aim at envisioned environments where devices are able to provide automatic services to users. Such characteristic must work without human intervention [1] [2] and must consider the necessary scalability to cope with an expected significant usage growth rate [7], while preserving users privacy [8]. Besides, the security model smart objects limita-

Hiro Gabriel Cerqueira Ferreira
Electrical Engineering Department, University of Brasilia UnB Campus Darcy Ribeiro Asa Norte Brasilia DF, Brazil, 70910-900.
E-mail: hiro.ferreira@gmail.com

Rafael Timoteo de Sousa Junior
Electrical Engineering Department, University of Brasilia UnB Campus Darcy Ribeiro Asa Norte Brasilia DF, Brazil, 70910-900.
E-mail: desousa@unb.br

tions 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 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 middleware and security technologies that can be used in IoT scenarios, considering that the middleware aims to allow things to communicate with internal and external entities while security measures provide privacy, authenticity, integrity and confidentiality to such communications.The technologies presented in the following subsections are correlated to the proposed security model in Section 4.

### 2.1 Related Security Technologies

The technologies briefly presented hereafter are commonly used within distributed scenarios such as World Wide Web (Internet), wireless sensors and actuators networks (WSAN), and Mobile Ad Hoc Networks (MANETs), which are precursors and enablers of IoT.

#### *2.1.1 AES-CCM*

AES-CCM is an operation mode of the Advanced Encryption Standard (AES). It uses counter mode added to cipher block chaining with message authentication code (CBC-MAC) [9] and is designed to provide privacy and authenticity in constrained environments and compact implementations [10]. Most WSANs use ZigBee [11], 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 [11], ZigBee with AES-CCM is suitable for IoT scenarios.

#### *2.1.2 TLS*

Transport Layer Security (TLS) is a channel-oriented protocol designed to secure data exchange between applications [12]. It uses symmetric (AES) and asymmetric (RSA) cryptography schemas to ensure privacy, authenticity and integrity in a communication session between client and server. HTTP over TLS, also known as HTTPS [13], is largely used by Internet services because it allows secure data exchange between HTTP servers and clients, such as web browsers and webpage 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 connections [14]. This way, it is also possible to keep interoperability throughout the World Wide Web.

#### *2.1.3 oAuth*

oAuth 2.0 is a low complexity authentication protocol that allows an entity to access resources belonging to other entities. This authorization has a limited scope, defined by the resource owner, through an issued access token [15]. This authentication and authorization suite is scalable and used by large enterprises (Google, Facebook, Windows live, GitHub and others) to authenticate multiple devices and applications. To such enterprises, generally speaking, oAuth manages applications access rights to users' resources [16] so that they only have access to what the user allows, for the right amount of time, without demanding 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 things. 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 owner, oAuth comes as a good candidate to solve authentication and authorization issues of IoT environments.

### 2.2 Related IoT Middleware and Architectures

The middleware and architectures here presented are intended to enable compliant objects to communicate with foreign entities, hence they should bring interoperability and security to these devices and foreign entities. Their components should be deployable, transparent [1], scalable and should be responsible for most processing and storage, so these middleware and architectures enable even severely constrained devices with no processing capabilities to be part of IoT environments.

This research intends to use existing solutions to serve as base for its experimentations and propositions. A middleware for IoT similar to described in above paragraph was sought and an overview will be presented on most relevant initiatives.

#### *2.2.1 LinkSmart*

The Hydra middleware, lately renamed to LinkSmart [17], was first designed to enable networking for embedded sys-

tems. It has evolved to a IoT like philosophy and now provides a deployable service-oriented architecture with embedded security.

Although such middleware accomplishes important milestones, it has some core issues to solve yet. To build LinkSmart devices it is necessary to load hardcoded scripts on LinkSmarts main software (proxies) based on Java to adapt devices to LinkSmart's open source protocols. Such middleware also demands processing capacity by devices, what makes it not simple to deployable for most legacy plain objects. It is somewhat exclusive. For those reasons this paper has not selected this proposal.

### *2.2.2 IoT-A*

IoT-A [18] is a research collaboration on IoT architectural reference model(ARM) funded by IETF under the ICT Theme of Framework Programme 7 (FP7). The research team managed to deliver a book and the ARM.

Unfortunately, such research does not provide a prototype, even-though 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, yet,as reference, it is very complete.

IoT-A's lack of a prototype has driven this work to search for another base middleware to serve as its base.

### *2.2.3 iCore*

iCore Project [19], also funded by IETF under the ICT Theme of Framework Programme 7 (FP7), lies in requirements gathering phase for future results. It has some implementations and promises to be scalable, embracing devices discovery service, eventing service and encompassing heterogeneity of devices.

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

### *2.2.4 UIoT*

UIoT was first proposed as a communication architecture to embrace legacy and later developed devices in [5] with use cases proposed in [6]. Its architectural model, represented in figure 1, provides uniform and transparent APIs to

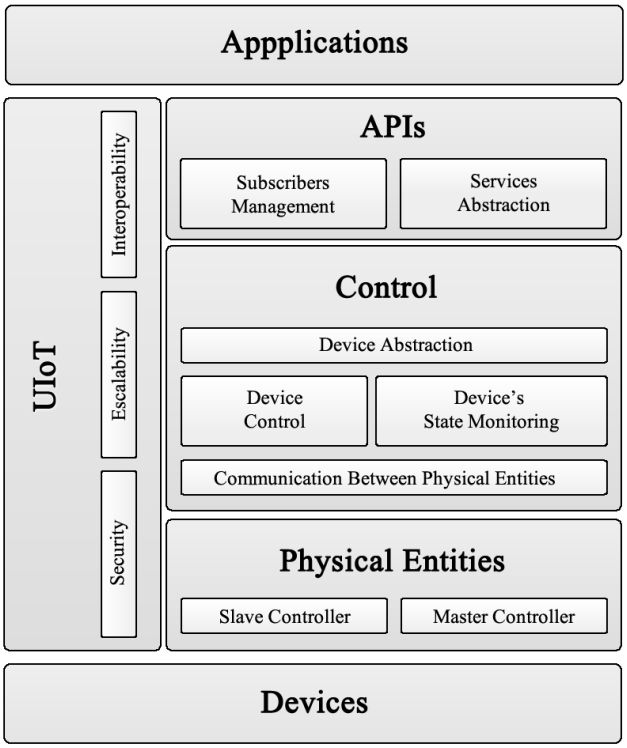


**Fig. 1** UIoT Architectural Model

applications, subscription services, concentrates administrative tasks inside the middleware and provides mobile serial digital and analog interfaces to communicate with end devices. UIoT abstracts devices complexity through UPnP device model [21], manageable by administrators through web forms and rest APIs. Despite had been developed in parallel to other presented architectures, UIoT has many similarities to them and 6LowPAN [27], but aiming on uniformity of services and reduction of load on end devices.

UIoT perform two routines to convert plain objects into smart objects or to aggregate smart objects under its uniform APIs: control routine and state monitoring routine. Control routine consists in executing requests on devices in order to change their state to achieve what applications desire, hiding operation complexity from them. State monitoring routine is intended to keep track of devices state in order to notify new state to states subscribers(eventing).

UIoT is internally composed by two types of entities: master controller and slave controllers. Master controller is responsible for keeping state, keeping abstractions and communicating with applications. Slave controllers are responsible for communicating with devices through its digital and analog interfaces. Those entities can vary from Raspberry Pi platform [22] to Arduino boards [23] or computer grids in cloud manner. They are scalable and should be chosen in order to feat the application scenario. Master controller must be able to abstract devices, hold services, data and con-

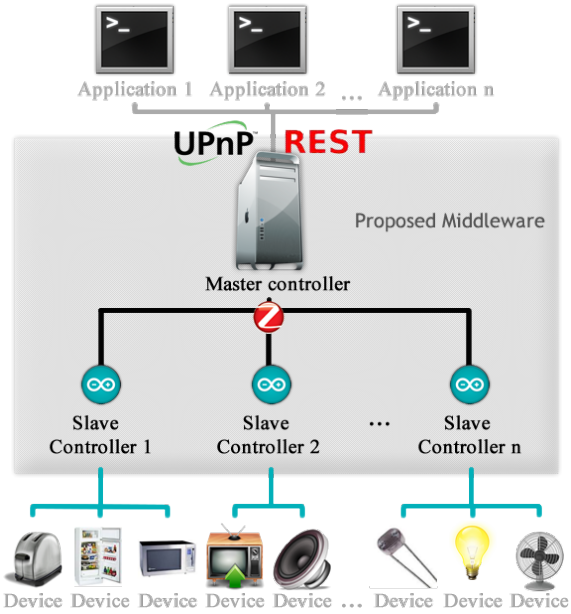


**Fig. 2** UIoT Prototype physical architecture

vert 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 zigbee mesh protocol [11].

UIoT's layered architecture allows new APIs to be added in Service Layer, new communication models, like Power-Line or bluetooth or WiFi, to be added in Communication layer and later developed devices to be added in execution layer. Since proposed transparent communication model is kept, it can be extensively extended to meet any scenario.

UIoT represents heterogeneous devices uniformly, allowing interoperability, has decoupled controllers, allowing mobility, and has prototype built with web technologies, allowing scalability.

The first prototype of UIoT was built with a set of existing technologies, such as UPnP, REST, ZigBee, Arduino and IP, in order to make it understandable and extensible by other researches. REST [20] and UPnP [21] APIs were provided for applications. ZigBee mesh communicate Master and Slave controllers. IP communicates Master Controller and applications. Arduino [23] with XBEE radio expansions represents slave controllers and parses master controller's instructions to end devices. This prototype was designed to cope with scalability, diversity, as well as devices computational and energy limitations. Figure 2 represents such first version.

UIoT yet requires security model. For that reason and other many advantages outlined in previous paragraphs, it was chosen as base middleware for this work.

## 3 Analysed Attack Models and Possible Defenses

With UIoT, we have defined the middleware. Now we need to define attack models to which we intend to protect it from. Based on [26], we have identified the following major vulnerabilities and attack models applicable to IoT: Passive Listening, Impersonation, Black Holes, Sybil Attacks,s Denial of Service and Physical Theft. Each one is introduced in following subsections.

### 3.1 Passive Listening

Entities listening to conversations between applications, controllers and devices without authorization caracterize 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. Any entity capable of listening tcp/ip network in promiscuous mode can trace interactions among master controller and applications. Any entity capable of listening to IEEE 802.15.4 defined radio channels can trace interactions among master controller and slave controllers. To be mitigated, this vulnerability requires private conversations among UIoT authentic entities.

### 3.2 Impersonation

Impersonations comprehend entities pretending to be 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 way to do this is the man-in-the-middle attack, in which an entity fakes to be the server for clients and the client for servers, thus, having access to both parts informations.

If an 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. It will be able to parse messaged between them, having access to all exchanged information, and making them believe there is no other entity in the middle. This vulnerability requires to uniquely identify UIoT authentic entities. 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 occupies privileged physical locations and joins controllers mesh network without authorization, faking being a part of such environment. It convinces its neighbors to drive their traffic

through it, them it does not forward received data, serving as a sink for such network.

To avoid this sort of attack, malicious entities should not be able to join UIoT networks. Authentication methods are required.

### 3.4 Sybil Attacks

In sybil attacks malicious entities uses UIoT in order to perform unauthorized actions. Faking a controller, 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 change messages in name of other devices or misleading control messages to networked controllers. Using passive listening, an attacker could get session details of applications and send messages in name of them.

In the same way as Black Holes, authentication methods are required. Also privacy and confidentiality in conversations during all times, including authentication, should be expected as a sybil node could get authentication credentials and change the packages to whatever it desires if channels are in plain text.

### 3.5 Distributed Denial of Services

Another concern is possible Distributed Denial of Services (DDoS) attacks. Malicious entities could perform, from different servers, a large amount of requests, making master controllers overload and not respond to any other request.

In order to prevent such attack, master controller should count and correlate received massages in order to filter and attend authentic requests while discarding DDoS classified messages.

## 4 Proposed Security Model

As described in section 2.1 above, AES, oAuth, and TLS are very well defined and accepted security technologies. They can be integrated to UIoT architecture so as to fulfill this architectures security requirement.

As described, the master controller has no computational or 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 by smart objects generated by our middleware above the plain objects.

This section covers the middleware internal and external security aspects, as well as authentication and smart objects access control list (ACL). Internal security comprehends communication between master and slave controllers, while external 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.

### 4.1 Internal Security

Slave controllers are supposed to be entities with few energy and computational resources so they must not handle too much policies and information processing tasks. This is the reason why this entity only deals with cryptography and integrity checks that comprise the ZigBee technology, which is already embedded on slave controllers [5].

ZigBees AES-CCM with 128 bits key answers this requirement. It allows secure communication by providing confidentiality with block cyphering, integrity with the message integrity code (MIC) and authenticity with the message authentication code (MAC) [10].

Configuring the ZigBee modules in the master and slave controllers so as to only accept connections with encryption enabled will provide security to the connection between these devices. 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.

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

### 4.2 External Security

Defined as the middleware most intelligent entity, the master controller centralizes most of the logical procedures and executes most of the hard processing tasks [5]. It is central middleware processor, the router between applications and slave controllers, as well as provider of abstractions of end devices.

Since it is the master controller 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 API [5] [6], both using HTTP. For that reason the master controller should perform its unicast communications over HTTPS [13] thus using TLS to enforce security, using AES and RSA to provide confidentiality, and MIC, MAC and SHA-1 provide data integrity. Prevention from DDoS can be done using one of the algorithms analyzed in Ref:ddos.

### 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.

Root users can create all the other types of users and manage all smart objects and their services. This special user is supposed to only manage 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` [5] and create, view, edit and delete all system users, all system roles, all slave controllers and all end devices with its 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 of 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 our middleware that 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 owners credentials, meaning to the smart object user that his username and password are safe. For that reason, the oAuths Implicit Authorization Grant [15] method 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

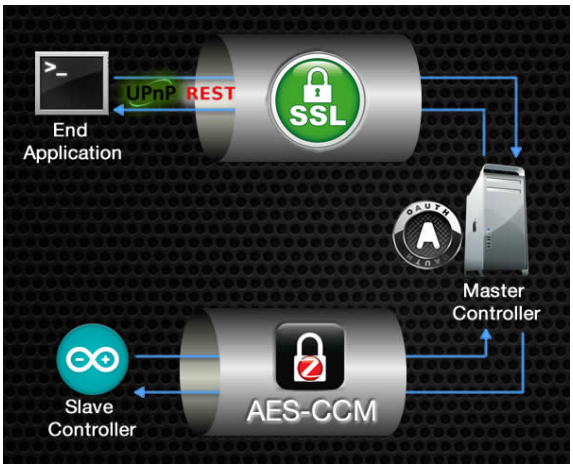


**Fig. 3** UIoT Standard Security Model

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. With this methodology, the master controller can keep track of what the application has being doing for further audits, if necessary.

With the specified user types, and the 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 show in early 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 [25].

The proposal of a first standard security model based on regular technology for UIoT is represented in figure 3.

## 5 Conclusions

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

UIoT proposed a transparent and deployable IoT middleware, with detailed use scenarios, and this paper analyses threats 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 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 low computational and energy resources.

It is interesting to note that, similarly to nodes in other networks such as ad hoc [29], IoT things can implement countermeasures to respond when they are attacked. These include even physically fighting back, an important subject in the area of cyber defense.

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 under the middlewares API, bringing uniformity to IoT's communication. Technologies such as DTMF, UPnP, Power Line and 6LoWPAN/CoAP allow devices to communicate, but all in 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** This work was supported by the Brazilian Ministry of Planning, Budget and Management, and the Brazilian research and innovation agencies CAPES and FINEP Grant RENASIC/PROTO 01.12.0555.00).

## References

1. Weiser, M. (1991). The Computer for the 21st Century. Scientific American, Vol.265, no 3, pp 94-104.
2. Giusto, D.; Iera, A.; Morabito, G.; Atzori, L. (Eds.) (2010). The Internet of Things, Springer. ISBN: 978-1-4419-1673-0.
3. Ashton, K. (2009). That 'Internet of Things' Thing. RFID Journal, www.rfidjournal.com/article/print/4986
4. Weber, R. H (2010). Internet of Things New security and privacy challenges, Computer Law & Security Report, Vol.26, Issue 1, pp 23-30, Elsevier.
5. Ferreira, H. G. C.; Canedo, E. D.; Sousa Junior, R. T. (2013). IoT Architecture to Enable Intercommunication Through REST API and UPnP Using IP, ZigBee and Arduino. 2013 IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), pp.53,60. doi: 10.1109/WiMOB.2013.6673340.
6. Ferreira, H. G. C.; Sousa Jnior, R. T.; Canedo, Edna D. (2014). A Ubiquitous Communication Architecture Integrating Transparent UPnP and REST APIs, International Journal of Embedded systems, Special issue on IEEE/IFIP EUC 2013 Embedded and Ubiquitous Computing, 13-15 Nov. 2013. Inderscience publishers.
7. Gubbi, J.; Buyya, R.; Marusic, S.; Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions, Future Generation Computer Systems 29 (2013) 1645-l60, Elsevier.
8. Atzori, L.; Iera, A.; Morabito, G. (2010). The Internet of Things: A survey, Computer Networks 54 (2010) 2787-2805, Elsevier.
9. Whiting, D.; Housley, R.; Ferguson, N. Counter with CBC-MAC (CCM), RFC-3610. Available at: <http://tools.ietf.org/html/rfc3610>accessed on Feb 18, 2014.
10. McGrew D.; Bailey D. AES-CCM Cipher Suites for Transport Layer Security (TLS), RFC-6655, July 2012. Available at: <https://tools.ietf.org/html/rfc6655>accessed on Feb 18, 2014.
11. ZigBee Alliance. Available at: <http://www.zigbee.org>accessed on Feb 18, 2014.
12. Dierks T.; Rescorlar E (2008). The Transport Layer Security (TLS) Protocol Version 1.2, RFC-5246. Available at: <http://tools.ietf.org/html/rfc5246>accessed on Feb 18, 2014.
13. Rescolar E. (2000). HTTP Over TLS, RFC2818. Available at: <https://tools.ietf.org/html/rfc2818>accessed on Feb 18, 2014.
14. Iwendi, C. O.; Allen, A. R. (2012). Enhanced security technique for wireless sensor network nodes. IET Conference on Wireless Sensor Systems (WSS 2012), pp.1,5, 18-19 June 2012. doi: 10.1049/cp.2012.0610
15. Hardt D. (2012). oAuth 2.0h Authorization Framework, RFC-6749. Available at <http://tools.ietf.org/html/rfc6749>accessed on Feb 18, 2014.
16. oAuth, open standard for authorization, 2014 Available at: <http://oauth.net/2/>accessed on Feb 18, 2014.
17. Hydra Middleware/Link Smart Middleware, 2014. Available at: <http://www.hydramiddleware.eu>accessed on Feb 18, 2014.
18. IoT-A, Internet of Things Architecture, 2014. Available at: <http://www.iot-a.eu>accessed on Feb 18, 2014.
19. iCore Project, 2014. Available at: <http://www.iot-icore.eu>accessed on Feb 18, 2014.
20. Stirbu,V.,"Towards a RESTful Plug and Play Experience in the Web of Things," Semantic Computing, 2008 IEEE International Conference on, pp.512-517, 4-7. Aug. 2008. doi: 10.1109/ICSC.2008.51
21. Universal Plug and Play Forum, Understanding Universal Plug and Play, 2000. Available at: <`http://www.upnp.org/download/UPNP_understandingUPNP.doc`>accessed on Feb 18, 2014.
22. Raspberry Pi, 2014. Available at: <http://www.raspberrypi.org/>accessed on Feb 18, 2014.
23. Arduino, 2014. Available at: <http://www.arduino.cc/>accessed on Feb 18, 2014.
24. JSON, 2014. Available at <http://www.json.org>, accessed on Feb 18, 2014.
25. Miorandi Daniele; Sicari, Sabrina; Pellegrini, Francesco De; Chlamtac Imrich; Internet of things: Vision, applications and research challenges, Ad Hoc Networks 10 (2012) 1497-1516. Elsevier, April 2012.
26. de Sousa Jnior, Rafael Timteo and Ricardo Staciarini Puttini. "Trust Management in Ad Hoc Networks." Trust Modeling and Management in Digital Environments: From Social Concept to System Development. IGI Global, 2010. 224-249. Web. 3 Nov. 2014. doi:10.4018/978-1-61520-682-7.ch010
27. Shelby, Zach, and Carsten Bormann. 6LoWPAN: The wireless embedded Internet. Vol. 43. John Wiley & Sons, 2011.
28. Mirkovic, Jelena, and Peter Reiher. "A taxonomy of DDoS attack and DDoS defense mechanisms." ACM SIGCOMM Computer Communication Review 34.2 (2004): 39-53.
29. Adnane, Asmaa, Christophe Bidan, and R. T. de Sousa. "Trust-based countermeasures for securing OLSR protocol." Computational Science and Engineering, 2009. CSE'09. International Conference on. Vol. 2. IEEE, 2009.