International Journal of Advance Research in Science and Engineering
Vol. No.6, Special Issue (01), September 2017, BVCNSCS 2017
www.ijarse.com
IJARSE
ISSN 2319 - 8354

AUTHORIZATION FOR INTERNET OF THINGS

D. Ramakrishna[1], D.Vasumathy Ramakrishna[2]
[1]*Computer Science Department, Bhavan's Vivekananda College of Science, Humanities and Commerece, Sainikpuri, Secunderabad*
[2]*Humanities and Science Department, St. Peter's Engineering College Maisammaguda, Opp.Forest Academy, Dhulapally, Near Kompally, Medchal (M), R.R.Dist., Hyderabad*

ABSTRACT: *This paper describes about the Authorization of Internet of Things. The architecture of Authorization of Internet of Things looks like this, the Device Owner is connected to Authorization Server and then it is connected to User by Internet. In this technique we require a framework for authorization for Internet of Things. The authorization is required at Authorization Server to secure the data, because the Authorization Server is a device which receives the information from Device Owner and then that is passed to the User. As there are lot threats to Internet, so we need to protect our data from the damage. To protect this data we can use encryption techniques like pre-shared keys or public key to authenticate the data at Authorization Server. In modern days all the devices are connected to Internet and application runs on those devices, so end User requires some information from the Device Owner, the Device Owner must communicate through Authorization Server, so it requires to their protect, in this case the Authorization Server requires a framework for Internet of things.*

Keywords: Device Owner, Authorization Server, Encryption, Pre-Shared Key and Public Key

I. INTRODUCTION

In this paper there are 2 terms used, first one is Authorization. Authorization means the access rights given resources related to security like computer security or information security in common and to access them in giving particular techniques. The second term Internet of Things, it is specified like the connection of inter-networking of physical devices, vehicles, building and other items, which are embedded with electronic devices likes actuators, sensors, software and network connectivity, which enables these objects to get the data and exchange information. Finally the paper title is Authorization for Internet of Things

The Authorization for Internet of Things is like providing security to Device Owner and end User, at Authorization Server. This is possible by using encryption techniques.

As we know that the whole world is connected to the Internet, so a lot of data transfer is done by Internet. The devices are used in globally accessible, handle very sensitive data or Novel business models need new access modes for accessing the data from Device Owners.

In this paper, we focus on the important concepts like security challenges, authorization and access control, in the context of Authorization Server providing the security to the Devices Owner and the end User.

II. RELATED WORK

Protecting the data is important in now days as there is a lot of the end user (U) who access the data using mobile, laptops or small portable devices. That's the reason we require to protect the data in this case, as the Internet is open world, we require a lot of security techniques to protect the data.

Access control lets only authorized users to access a resource, such as a file, Internet of Things device, sensor or URL. All modern operating systems limit access to the file system based on the user. For instance, the super user has wider access to files and system resources than regular users.

In this case the Internet of Things context, access control is needed to make sure that only trusted parties can update device software, access sensor data or command the actuators like to perform a given operation. The business models like as Sensors As a Service (SAS), which helps to solve the access control the data ownership issues, where they might for instance like sell a sensor like whether temperature sensor data to the clients.

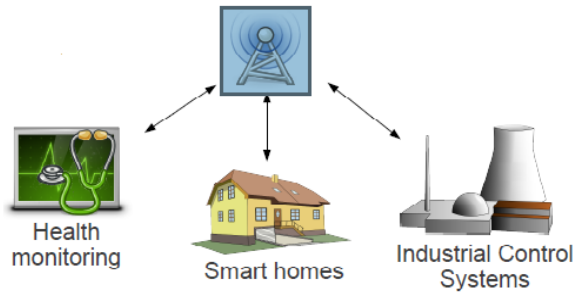

Figure 1: Connection of Internet of Things

Figure 1 explain about the connection of Internet of Things are connected to the Health Monitoring, Smart Homes, Industrial Control Systems and etc,. Where the authorization is required.

III. REQUIREMENTS

Differentiated access control rules for different requesting users: Local enforcement of certain conditions (e.g. on device-state, position, time). Minimal communication requirements and low computational overhead. Protect access control information itself Dependent on a minimum of other functions. End-to-End protection of protocol messages.[1]

IV. ARCHITECTURE

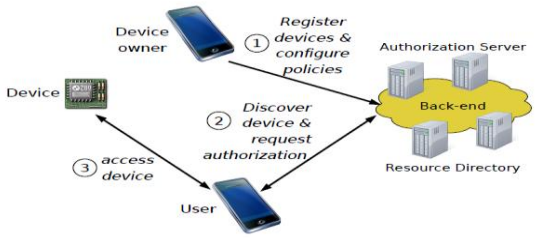

Figure 2: Authorization Framework Architecture

In figure 2, the Device Owner first register devices and configure polices (policies are based on local conditions e.g. state of the device, time, position), at the Authorization Server (A.S.) and Resource Directory, which also known as Back-End. The Authorization Server discovers device by User and User request authorization from Authorization Server. The User then accesses (The access control solution shall be dependent on a minimum of other functions.) the device services.[1]

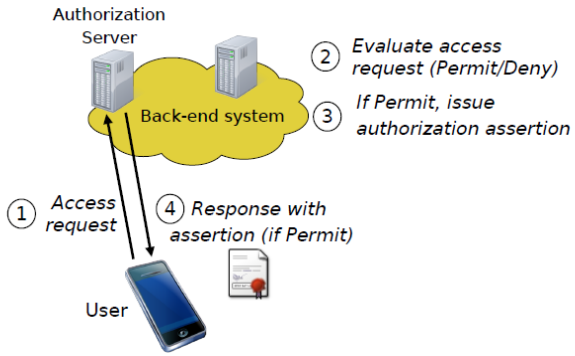

Figure 3: Authorization Access Procedure

Figure 3 explains authorization access procedure. In this first User request for access permission to Authorization Server (Back-End System). The Authorization Server does the following tasks, as it gets the request. It evaluates access requests and checks whether to Permit or Deny the request by the User (U). If Authorization Server (AS) Permit, then it issues an authorization assertion to the User (U). The User (U) gets a response with assertion if it is Permitted response.[1]

V. FRAMEWORK FOR AUTHORIZATION

The requirement to fulfill the fine-grained access control, we have to select the access control standard like XACML (eXtensible Access Control Markup Language) [2], since it is more predominant standard in the area and has been used in industry to some extent.

The process of evaluating XACML (eXtensible Access Control Markup Language) policies is too heavy weight for constrained devices in the framework, therefore we can propose to externalize the most of the authorization decision process (All introduced security mechanisms shall be designed such that the total overhead due to computation and especially communication is as low as possible on the device side) and have the device perform primarily authorization enforcement (The solution shall provide end-to-end protection (integrity and confidentiality) of relevant parts of the protocol messages, as well as replay protection).

The process to deal with local conditions affecting the access control decision (The framework shall support policies based on local condition example, like state of device, time, position), other information about local conditions must be transported to an external policy decision point or some access control decision be made within the constrained device. It may later be preferred for several reasons like: transporting information about local conditions for each policy decision introduces delays and adds a transmission cost to the device and moreover, the local conditions may have changed at the time of enforcement. Furthermore, we can express the local conditions as XACML (eXtensible Access Control Markup Language) Obligations i.e. the conditions which makes a decision as valid can be an external authorization decision in the framework.

In the sequence to convey the authorization decisions from the external decision point to the device, we can opt to use assertions, which are digitally signed data objects containing asserted information and in particular way we use SAML authorization decision assertions[3] as a template. An alternative would have been to use OAuth access tokens[4] as starting point, and at the end result would have been similar, but we choose SAML since it is well integrated with XACML (eXtensible Access Control Markup Language).

We have three main elements in our framework, they are like a Device (D) hosting resources, the next element is User (U) wishing to access a resource and the last element is Authorization Server (AS) located in the back-end, which performs policy evaluation and issues authorization assertions for resource access to the User. The User sends these assertions to the device along with the request. The Authorization Server (AS) is acting on behalf of a device owner who has configured the resource access policies.

Independent of the authorization mechanism, we also need a fourth entity that facilitates resource discovery, a Resource Directory (RD) maintains descriptions of resources. Our design extends the use of the directory to also manage secure data, relevant to devices (example device public key, the capability to process local conditions). It must be noticed that privacy reasons, that the resource descriptions may also be subject to access control (The authorization framework shall support secure access to access control related information), and should not be transmitted over an unprotected channel.

It is straightforward to apply our access controlframework to this data. The resulting architecture is explained in figure 2.

In this architecture, the authorization framework requires the following minimal set of the following functions according the given consideration: The access control solution shall be dependent on a minimum of other functions. They are as follows:

i. The Authorization Server (AS) must to be able to bind the

User to the assertion. If the authorization decision depends on the User's identity the Authorization Server (AS) also requires to authenticate the User. In cases where the User's identity is not relevant (example like purchasing service) as alias can be used instead. The bonding can be achieved by including the public key (security key) or the alias of the User in the assertion. The alias can be authenticated using the scheme.

ii. The Device (D) must be able to verify that an assertion is valid and from a trusted source. To achieve this the Authorization Server (AS) needs to sign the message using a key that is known to and trusted by the Device (D).

iii. The Device (D) must be able to bond with the User (U) to the assertion. This can be achieved by implicit or explicit authentication of the User (U) (or the alias used)

To compile with security mechanisms shall be designed such that the total overhead due to computation and especially communication are as low as possible on the device side. All introduced security mechanisms shall be designed such that the total overhead due to computation and especially communication is as low as possible on the device side, the protocols are used to implement these functions which should be used as minimum of message exchanges with the Device (D), this ideally not more than if the Device (D) accessed without authorization mechanisms.

The transport protocol can be build upon the IETF Constrained Application Protocol draft (CoAP)[5], adding security information to the CoAP message where needed. CoAP is specifically designed for constrained devices and features a very low overhead compared to e.g. HTTP (Hypertext Transfer Protocol), nevertheless our framework is not specific to CoAP and would also work with other application layer protocols.

The given framework describes above that we can implement functions complying with all our requirements and additional information even to fulfill those of Naedele.[6]

VI. KEY CONCERN IN FRAMEWORK

The requirement for solution shall provide end-to-end protection (integrity and confidentiality) of relevant parts of the protocol messages, as well as replay protection. This assumes a key establishment procedure, and unless keys are provisioned, this has in turn been predated by an authentication or security procedure. In the above authorization of framework, we neither require a key agreement procedure nor a particular authentication protocol, but we have to nevertheless accountable for keys that are established as that impacts, what capacity left in the Device (D) for security related tasks. We must be limited to ourselves on to two main candidates here.

The best option suitable for CoAP as straightforward is DTLS [7] based on raw keys (Public or Pre-Shared), in which case the DTLS record protocol provides cryptography, integrity and replay protection of CoAP messages. For extremely controlled devices, however, the DTLS handshake may impose a considerable setup time.

The modeled framework of an object authentication based approach. In this approach we can use symmetric keys for an object protection, but works with both symmetric and asymmetric established keys: Assume first that the Device (D) and Authorization Server (AS) have established each other's public keys. By including a verified public key of the user (i.e. it is obtained by the Authorization Server (AS) in the assertion request) in the assertion and once in the payload the User (U) and Device (D) can perform the analog calculations and derive a Symmetric Key.

Now lest assume Instead that Device (D) and Authorization Server (AS) have established shared symmetric keys. A unique User (U) alias instead of public key in the assertion, The Device (D) and the Authorization Server (AS) can use a suitable one-way key derivation function to derive a symmetric key.

VII. PROCEDURES IN AUTHORIZATION FRAMEWORK

We require a set of procedures and protocols to perform the following:

A. Device owners registering new devices and their relevant authentication data.
B. Users (U) finding a device and requesting an authorization assertion for it.
C. User (U) accessing a device using a previously obtained authorization assertion.

To comply with our requirements, the security mechanisms shall be designed such that the total overhead due to computation and especially communication is as low as possible on the device side. We design our protocols such that they do not require additional message exchanges compared to unprotected CoAP exchanges.

A. Registering a New Devices (D)

As we assume the existence of a resource directory such as the IETF Resource Directory[8], which supports procedures for the device to initiate registration of resource description to the directory. We assume that security relevant data for a device such as its public key, the Authorization Server it trusts, its owner, and the Obligations which it can process, may be registered as device meta-data in the directory, and can be queried by relevant entities. Publishing this meta-data can follow the same procedure as for the publication of the device's resources.

B. Getting an Authorization Assertion

The best way to access a resource on a Device (D), the User (U) needs not only to find the Uniform Resource Identification of the resource, but we also to acquire an authorization assertion and a cryptographic key to use in security protocols with the Device (D). The Uniform Resource Identification and Device connected to the authentication parameters can be retrieved from the Resource Directory.

Among these is the address of the Authorization Server (AS) to be trusted by the Device (D). The User (U) requests an assertion to access a particular resource from this Authorization Server (AS), which internally runs the XACML ((eXtensible Access Control Markup Language).) request-response protocol to find out if the User (U) is granted access. The Authorization Server (AS) returns an assertion and a Device Key to the User (U). Depending on whether asymmetric or symmetric keys are used, the assertion contains either a public key or a unique alias of the User (U).

C. Accessing a Device (D)

The resource on the Device (D) and the User (U) now sends a CoAP request including the assertion to the Device (D), secured with a protocol/crypto suite supported by the Device (D). CoAP supports the use of optional request information to be carried as a CoAP Option interspersed between header and payload. We propose to introduce an Assertion Option in CoAP. Furthermore, in our object security approach we replaced the CoAP payloads with object secured equivalents based on the Device Key obtained from the Authorization Server.

The Device (D) verifies the assertion, matches the access rights authorized in the assertion with the actual access request, and verify the local conditions (if any) specified in the assertion.

If all verifications are successful the request is granted with consequential processing and response. Replay protection is provided by giving the assertions a short, pre-defined validity time, and storing on the device a list of recently used assertion identifiers.

The DTLS offers bundled decoded and integrity protection of both headers and payload, the main goal or object of authorization approach allows for a trade-off between protection against performance. The assertion, payload and trust model depending may need to be decoded because of overhearing will reveal information about the User,s (U) request, which may be privacy sensitive. Wrapping the payloads as secure objects allow differentiated protection of the content based on its sensitiveness.

For example, in a CoAP GET request, the assertion could be integrity protected only, while the response payload would be encrypted and integrity protected. The assertion and request payload in a CoAP PUT / POST, would be integrity proctected and the response would be unprotected.

VIII. AUTHORIZATION SERVER

The Authorization Server (AS) consists of two components, they are i) an assertion issuing system and ii) an access control system. The assertion issuing system encodes the authorization decision as an assertion. The access control system produces policy-based access control decisions using XACML(eXtensible Access Control Markup Language). How the policies are created and administrated is out of scope for this paper. When a User is granted access by the access control system.

It is possible that the access granted by such an assertion depends on parameters known only to the Device (D), in which case the Device (D) will evaluate those and grant or deny access based on the outcome of this evaluation. This means that at least some devices will perform more than pure enforcement of access control decisions.

The authorization decision which enables the Device (D) to enforce, the assertion needs to provide the following information, they are follows:

- Which resource does the decision applies for.
- Which action (GET, PUT, POST, DELETE) does the decision apply for.
- Which subject does the decision apply for, and how can this subject be authenticated (if necessary).
- Which assertion server has issued this assertion (this information might be implicit from the signature of the assertion).
- Under which other conditions is the assertion valid (expiration date, replay protection, parameters evaluated by the device at access time).

Since the full syntax of XACML (eXtensible Access Control Markup Language) Responses and SAML Assertions includes a large number of features, we have defined a subset of both standards, in order to simplify the processing on the Device (D). Furthermore the XML representation of this subset is too verbose for efficient transmission over limited channels, therefore we have defined a compact JSON-based notation for our SAML and XACML (eXtensible Access Control Markup Language) subset. This approach reduces the size of the assertion roughly by a factor of ten.

IX. IMPLEMENTATION

Let the framework have Device (D) as part of it which is implemented using the object authentication based approach and symmetric keys for an example, let us take platform for it which has the following details: The Arduino Mega 2560 board3. This board features a 16 MHz processor, 256 kB of Flash Memory, 8 kB of SRAM, and 4 kB of EEPROM. We chose this board in order to test our approach on the low end of the performance spectrum for target constrained devices.

The board was programmed in C using a custom implementation of the CoAP protocol stack, the Cryptosuite library for HMAC-SHA256 and an optimization of the 8-bit AES implementation by Brian Gladman Processing the CoAP messages on the device, including our authorization handling, requires roughly 7.3 kB of static memory (including Arduino internals such as UDP, Ethernet, SPI libraries, etc), which places us close to the upper limit of what this board can do.

From the required operations the most time consuming ones unsurprisingly turned out to be encrypting, decrypting, integrity protection, and integrity verification. Other operations such as matching the assertion to the requested action turned out to consume only negligible time.

We chose to use the IETF JSON Web Encryption (JWE) [9], an emerging secure object standard, for wrapping the assertion and payload. Note that this wrapping expands the payload size drastically. For example a typical sensor reading could be a 4-byte integer. If that would be protected by AES encryption and a HMAC message authentication code, we would have 128 bytes of encrypted text due to the block-size padding and another 160 bytes for the MAC.

X. SECURITY EVALUATION

In the present framework, we aim to protect the following assets: The data on devices, the devices themselves, and the services offered by devices.

Our measures to protect these assets are to enforce fine grained restrictions on accessing the devices (as opposed an all-or-nothing approach,that would just require authentication). Due to the setup of our framework, we also need to protect authorization decisions, the authorization policies, and relevant attributes to make these decisions.

Note that only the protection of authorization decisions needs to be verified on the Device (D), everything else is performed on more powerful back-end machines.

The Authorization Server (AS) is a Trusted Third Party from the point of view of the Device Owner, which if compromised could e.g. issue assertions to unauthorized parties or use a derived key to decrypt an eavesdropped GET response.

The end-to-end security setting has two sides. Since all data is verified and protected in the Device (D) there are no intermediary attack targets for breaking confidentiality or integrity. But also since the Device (D) is in principle open for access from arbitrary users, additional overload protection mechanisms may be needed, e.g. external firewall functionality restricting the number of simultaneous requests and/or verifying assertions before forwarding . An alternative approach is to use a gateway that has full, direct access to the devices it manages, and filters access requests based on its access control policies. Such an approach has the advantage that all authorization handling is moved to an entity without the resource constraints present on the devices. However one disadvantage is that we cannot maintain end-to-end protection of the protocol messages, since the gateway needs to be able to read them. Thus privacy critical requests cannot be protected should the User (U) distrust the gateway. Furthermore this approach is not applicable to a scenario featuring devices only locally accessible in isolated places.

XI. CONCLUSION AND FUTURE WORK

We have presented a generic authorization framework for Internet of Things devices built upon existing Internet and access control standards supporting fine-grained and flexible access control to constrained devices.

The key components in this framework are the Authorization Server (AS) and our newly designed assertion profile defined as subset of SAML and XACML (eXtensible Access Control Markup Language) and compactly represented in a JSON notation. Of special significance we used XACML (eXtensible Access Control Markup Language) obligations to enable any kind of local decisions in the device. Supporting components are the extension of the Resource Directory for publication of Device (D) capabilities for local decisions and enforcement, and key management procedures used to establish security between the Device(D) and the Authorization Server (AS) / User (U).

Performance critical parts of this framework have been implemented and tested using a object security based approach on an example Device (D) and thereby shown that the authorization procedures can be executed in a reasonable time-frame on certain classes of constrained Devices (D). The security evaluation elucidates the trade-offs and assumptions that where made for this framework and specifies which security assurances the framework provides.

The use of JWE as wrapper format for secure object is suitable for the assertion but highly non-optimal for payloads of a few bytes, which are common in CoAP. Both the JWE header and the crypto payload could be made more compact for this kind of deployment. Potential future work include exploring and standardizing the use of stream-ciphers and MAC for JWE. Other topics for standardization are our assertion profile of SAML and XACML (eXtensible Access Control Markup Language), and device registration of security related meta-data using the Resource Directory.

XII. REFERENCES

[1] Ludwig Seitz, Goran Selander, Christian Gehrmann, “Authorization Framework for the Internet-of-Things”, IEEE 14th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM). pp. 1-6, 2013.

[2] S. Godik, and T. Moses (eds.), “eXtensible Access Control Markup Language (XACML),” Organisation for the Advancement of Structured Information Standards (OASIS), Standard Version 2.0, February 2005.[Online]. Available: http://www.oasis-open.org/committees/xacml

[3] E. Rissanen, and H. Lockhart (eds.), “SAML 2.0 Profile of XACML Version 2.0,” Organization for the Advancement of Structured Information Standards (OASIS), Committee Specification, August 2010, http://www.oasis-open.org/committees/xacml.

[4] L. Daigle and O. Kolkman, “The OAuth 2.0 Authorization Framework,” Internet Engineering Task Force (IETF), Request For Comments (RFC) 6749, October 2012, http://www.ietf.org/rfc/rfc6749.txt.

[5] Z. Shelby, K. Hartke, and C. Bormann, “Constrained Application Protocol (CoAP),” Internet Engineering Task Force, Internet-Draft draftietf-core-coap-14, March 2013, work in progress.

[6] M. Naedele, “An Access Control Protocol for Embedded

Devices," in Proceedings of the fourth IEEE Conference on Industrial Informatics, INDIN. Singapore: IEEE, August 2006, pp. 565–596.

[7] E. Rescorla and N. Modadugu, "Datagram Transport Layer Security Version 1.2," Internet Engineering Task Force (IETF), Request For Comments (RFC) 6347, January 2012, http://www.ietf.org/rfc/rfc6347.txt.

[8] Z. Shelby, S. Krco, and C. Bormann, "CoRE Resource Directory," Internet Engineering Task Force, Internet-Draft draft-shelby-core-resourcedirectory-05, February 2013, work in progress.

[9] M. Jones, E. Rescorla, and J. Hildebrand, "JSON Web Encryption (JWE)," Internet Engineering Task Force (IETF), Internet-Draft draftietf-jose-json-web-encryption-08, December 2012, work in progress.