Galileo 2016-2017
Project G16_31 "Security protocols for the Cloud-oriented Internet of Things (SeCIoT)"

Object: Final Scientific Report

Description of the activity

With good adherence to the original project plan, the SeCIoT research activities have been organized in the following phases (Mi stands for i-th month):

M1-M3: Study of the state of the art and periodic brainstorming sessions
M3-M9: Design of the novel IoT/Cloud security protocols and mechanisms as well as Cloud-oriented IoT applications
M3-M12: Implementation, integration, and validation
M8-M12: Dissemination

Preliminary activity

In January, both teams started to study the state of the art related to secured IoT systems supported by the Cloud. This preparatory activity served to prepare the first project meeting.

1st Project Meeting (Grenoble 6/2/2017)

The first project meeting took place in Grenoble, at LIG Lab, on Monday the 6th of February. The Italian team was represented by Michele Amoretti and Francesco Zanichelli (plus Gianluigi Ferrari connected via Skype).

The French team presented its activities related to IoT and security

- DataTweet project
- scalable access method for LoRA, SIGFOX, 5G MTC
- WalT platform: http://walt.forge.imag.fr

The Italian team illustrated its activities related to cloud, IoT, and security

- CloudAWM
- ADGT
- Distributed AES

Among the topics that could have been developed by joining forces, the teams agreed to start with the implementation of the OSCAR architecture (presented by the French team in the paper "OSCAR: Object Security Architecture for the Internet of Things" [1]), because of the OSCAR's good fit with the OpenStack-based cloud system deployed by the Italian team at its Department premises. During the meeting, it was also decided to create a GitLab repository for sharing reference papers, project reports (as wiki documents) and source code.

Web page and GitLab setup

After the first meeting, the Italian team created a web page for the SeCIoT project (http://dsg.ce.unipr.it/seciot), while the French team created the GitLab repository (https://gitlab.imag.fr/SeCIoT/).

OSCAR implementation (part 1)

In the four months following the first project meeting, both teams worked on implementing the OSCAR security architecture [1] in which the main actor is the Authorization Server (AS), a trusted entity that

1. stores certificates of Producers (constrained CoAP nodes that provide data in the form of signed and encrypted resource representations);
2. receives subscriptions of Producers for generated resources;
3. generates and provides Access Secrets protecting Producer resource representations.

When a Consumer (CoAP client) requests access to a Producer resource (e.g., a sensor), AS returns the Access Secret that allows the Consumer to get a resource representation from the Producer. The Access Secret is a token from which a Producer derives a symmetric encryption key to encrypt a resource representation. Producers/Consumers and AS use a secure DTLS session to exchange Access Secrets and certificates.

The implemented AS is based on "Canopus" [2], a basic server written in Go language, and supporting DTLS with OpenSSL API v1.1.0e. The following features have been added:

- Private Key
- Public key in X.509 Certificate
- Exchange of X.509 certificates
- Certificate Authority for verifying certificates
- MySQL Database for saving Access Secrets, resources and certificates of Concumers
- Server private key and certificate, as well as CA certificate (created with the certtool of GnuTLS)
- Concurrency (by means of Goroutines)

Extensive tests proved the correctness and robustness of the implemented AS.

2nd Project Meeting (Parma 8/6/2017)

The second project meeting took place in Parma, on Thursday the 8th of June, in the Department of Engineering and Architecture. The French team was represented by Andrzej Duda.

Both teams presented and discussed the ongoing implementation of the OSCAR architecture. The French team suggested to implement and test Producer nodes for different resource-constrained devices. Everybody agreed to write a paper on the OSCAR-related activity. The French team also presented novel ideas on using blockchain for secured data sharing in IoT environments.

OSCAR implementation (part 2)

After the second meeting, both teams worked to complete the implementation of the OSCAR security architecture.

In particular, a Producer was implemented, based on "FreeCoAP" [3], thus written in C language. In such a Producer, DTLS is supported with GnuTLS API (v3.5.3), keys are derived with MD5, data is encrypted using AES-128 and signatures are performed with SHA256withECDSA. The Producer is characterized by the following behavior:

- Setup parameters of DTLS connection and make DTLS handshake
- Send request to server
- Receive response from server (the received response is formed by access secret and access secret ID necessary to derive the key)

- Listen to incoming connections of Consumers
- Derive key and encrypt data
- Send encrypted and signed data

The Producer code was successfully tested on Raspberry Pi 2 devices.

Later, a Consumer based on the "Californium" project [4] was developed in Java language. In such a Consumer, DTLS is supported with Scandium API, keys are derived with MD5, data is encrypted using AES-128 and signatures are performed with SHA256withECDSA. The Consumer is characterized by the following behavior:

- Setup parameters of DTLS connection and make DTLS handshake
- Send request to server
- Receive response from server (the received response is formed by access secret and access secret ID necessary to derive the key)
- Send request to Producer
- Receive message from producer, derive key, descrypt data and verify signature

A "many Producers for one Consumer" scenario was implemented and tested.

Finally, a Proxy Server was implemented and deployed on the OpenStack-based cloud installed at the Department of Engineering and Architecture of the University of Parma. This facility includes one master server and several
slave servers running a dynamic set of virtual machines (VMs). According to the OSCAR security architecture, the Proxy Server stores the resources in an encrypted form when the Producers are highly constrained. When the protected resources are published to the Proxy Server, they need to be protected with double encryption mechanism [1].

3rd Project Meeting (Grenoble 17/7/2017)

In July, Michele Amoretti spent one whole week in Grenoble, to work with the French team on the project. The week started with a project meeting, where Michele presented the current status of the implementation of the OSCAR architecture: Authorization Server, Producer and Consumer prototypes available in shared repositories.
The French team made some insightful remarks and proposed to use OSCOAP and OSCON for Producer-Consumer interaction when Producers have to run on constrained devices with duty cycling, in order to overcome the problem of signature generation.
As a novel direction, it was decided to study how to substitute the Authorization Server with a more robust, distributed solution, in order to avoid the single-point-of-failure issue.
The French team proposed to investigate the use of a blockchain-like ledger, consensus-based.
During the week, the teams produced the first draft of an architecture based on blockchain-like ledger and group key distribution, denoted as IoTChain. It was also planned to write a paper to be submitted to the IEEE Wireless Communications and Networking Conference (WCNC), 2018 edition. WCNC is the major IEEE conference on wireless research, technology, and applications.

Definition of the IoTChain architecture

IoTChain is a combination of the OSCAR architecture and the ACE authorization framework [5] to provide an E2E solution for the secure authorized access to IoT resources. IoTChain consists of two components, an authorization blockchain based on the ACE framework and the OSCAR object security model, extended with a group key scheme. To avoid confusion between OSCAR and ACE taxonomies, the ACE one is always adopted in IoTChain. Thus, OSCAR's Consumer and Producer are named Client and Resource Owner, respectively.

The blockchain provides a flexible and trustless way to handle authorization (based on smart contracts published by resource owners), while OSCAR uses the public ledger to set up multicast groups for authorized clients.

IoTChain implementation (part 1)

To evaluate the feasibility of the IoTChain architecture, we have implemented the authorization blockchain on top of a private Ethereum network. We executed several experiments that assess the performance of different architecture components.

The description of the IoTChain architecture and the preliminary experiments have been summarized in a paper submitted to the IEEE WCNC 2018 conference. Later, the paper was accepted for the main track of the conference [6]. It will be presented at the conference - taking place in Barcelona next April - by Michele Amoretti.

4th Project Meeting (Grenoble 24/11/2017)

On Friday the 4th of November, Michele Amoretti joined the French team for a project meeting at LIG Lab, in Grenoble. Both teams discussed the ongoing work on the IoTChain implementation, focusing on the role of Smart Contracts as authorization policy enablers.

IoTChain implementation (part 2)

From Monday the 27th of November to friday the 1st of December, Francesco Medioli joined the French team in Grenoble, to work on the IoTChain implementation. Two weeks later, Timothy Claeys joined the Italian team in Parma for the same reason.

With respect to the IoTChain architecture described in the WCNC paper [6], the one resulting from the joint work of the research teams during this last period of the project is characterized by a new entity, namely the Public Authority (e.g., a municipality). It is a trusted third party that communicates with the Client and the Resource Owner and is responsible for deploying a smart contract on the blockchain to store the ethereum address and validate the identity of Clients and Resource Owners.

The IoTChain authorization flow is characterized by the following phases:

1. The Public Authority publishes the first smart contract (denoted as PA) on the blockchain. The PA smart contract exposes a function to add/remove an ethereum address of a Client or a Resource Owner to a structure (mapping) with a modifier. Only the creator of the smart contract can access these functions.
2. The Resource Owner communicates its resources to the Public Authority and what kind of the operation the PA has to perform in order to validate an identity of a Client (e.g., "for these resources, check that the Client lives in that street").
3. The Client (through the Android App) communicates his/her credentials (e.g., driver license) to the Public Authority to register the Client's data and the ethereum address.
4. The Public Authority adds, through a function of the smart contract, the ethereum address of the Client to the mapping. The Public Authority has to know if a certain address can ask or not for the token and decides whether to add or not a Client. In this phase the PA can add the ethereum addresses of the validate Resource Owners too.
5. The Resource Owner creates the second smart contract, which can interact with the first smart contract deployed by the Public Authority through an abstract contract. This contract

has the only purpose to declare the function of the first smart contract, without the implementation. In this way it may be possible to check if a certain ethereum address is included into the mapping. The contract checks if the address matches a "true" value. Moreover, in this phase (if the IoT device was not yet installed when the RO had interacted with the PA) the RO can add one of its resource for a specific Client.

6. The Client requires the execution of the second smart contract. If the ethereum address of the sender for a specific resource is included into the mapping, the contract generates an access token for the Client for the specific resource. Plus, the "LoginAttempt" event of the contract is triggered.

The ramaining phases are the same of the original IoTChain architecture (as described in the WCNC paper [6]). The Client requests the encryption keys necessary to decrypt the resources from the Key Server, after a challenge-response. The Key Server checks, through an event, if the Client's token exist in the blockchain (this part is explained in details in the "App & Server" section). Then, the Client receives a personal key and takes part in the self-healing group key distribution process. Finally, the Client can download the encrypted resources from the Proxy Server.

5th Project Meeting (Parma 13/12/2017)

At the end of the week he spent in Parma to collaborate in the implementation of the IoTChain architecture, Timothy Claeys represented the French team at the final project meeting. Here, the achieved results were discussed and plans for future work were made.

In particular, performance evaluation tests were planned, in order to complete the validation of the implemented IoTChain architecture and start writing a new paper.

Achieved objectives

With respect to the project plan, all four objectives have been met:

- Energy-efficient mechanisms for data security – The **OSCAR architecture** provides a novel scheme for IoT security based on data payload protection, in order to guarantee confidentiality and authentication at several levels, allowing plain or partial access to the message content based on groups of users.
- Energy-efficient schemes for secure data communication – The **IoTChain architecture** enables security of communications suitable for highly constrained nodes.
- Energy-efficient algorithms for secure cloud services – The **OSCAR architecture** includes a solution based on double signatures so that the IoT data can be stored in the Cloud in an encrypted form and only final consumers have access to the cleartext form. Moreover, the **IoTChain architecture** provides blockchain-based authorization mechanisms for excluding malicious actors.
- Integration and implementation of the proposed protocols on experimental platforms. Both the **OSCAR architecture** and **IoTChain architecture** have been implemented and validated on experimental platforms based on COTS technologies, OpenStack (concerning cloud services) and Ethereum (concerning the blockchain).

References

[1] M. Vucinic, B. Tourancheau, F. Rousseau, A. Duda, L. Damon, and R. Guizzetti, "OSCAR: Object Security Architecture for the Internet of Things," Ad Hoc Networks, vol. 32, pp. 3 – 16, 2015.

[2] Canopus project. [Online]. Available: https://github.com/zubairhamed/canopus

[3] FreeCoAP project. [Online]. Available: https://github.com/keith-cullen/FreeCoAP

[4] Californium project. [Online]. Available: https://eclipse.org/californium/

[5] L. Seitz, G. Selander, E. Wahlstroem, S. Erdtman, and H. Tschofenig, "Authentication and Authorization for Constrained Environments (ACE)," Internet Engineering Task Force, Internet-Draft draft-ietf-aceoauth-authz-07, August 2017, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-ace-oauth-authz-07

[6] O. Alphand, M. Amoretti, T. Claeys, S. Dall'Asta, A. Duda, G. Ferrari, F. Rousseau, B. Tourancheau, L. Veltri, F. Zanichelli, "IoTChain: A Blockchain Security Architecture for the Internet of Things", IEEE WCNC 2018, Barcelona, Spain, 15-19 April 2018. Accepted for oral presentation and publication in the conference proceedings.

Parma, 12/2/2018

Michele Amoretti
Italian Team Coordinator