Towards a standardized identity federation for Internet of Things in 5G Networks

Bernardo Santos
OsloMet – Oslo Metropolitan University
Oslo, Norway
bersan@oslomet.no

Van Thuan Do
Wolffia AS
Fornebu, Norway
vt.do@wolffia.no

Boning Feng
OsloMet – Oslo Metropolitan University
Oslo, Norway
boning.feng@oslomet.no

Thanh van Do
Telenor Research & OsloMet – Oslo Metropolitan University
Fornebu, Norway
thanh-van.do@telenor.com

***Abstract*—With the upcoming introduction of 5G networks for our daily usage and convenience and with the purpose of accommodating a countless amount of Internet of Things (IoT) devices and applications alongside the billion devices that already use the network (e.g. mobile phones), one of the key aspects for its success is to ensure to these upcoming devices enhanced but affordable security. With this paper, we present a solution that introduces an Identity Federation mechanism that reuses the SIM authentication for cellular IoT devices, enabling single sign-on features. This solution aims to relieve the responsibility of IoT providers for developing (proprietary) device identity management mechanisms while allowing to reduce operation costs overall.**

Keywords—mobile identity management, cross layer identity federation, mobile network security, IoT security, cyber security, cross layer security

I. INTRODUCTION

The term Internet of Things (**IoT**) was coined by Kevin Ashton of Procter & Gamble and later on in MIT's Auto-ID Center in 1999, however only in past few years the term has become widely used since its growth is at an incredible rate with both the number and variety of devices connected to the Internet are increasing at a very accelerating pace. To meet these devices' demands, the mobile industry applies tremendous effort towards the application of **IoT** technologies such as Extended Coverage GSM for Internet of Things (**EC-GSM-IoT**), Long Term Evolution Machine Type Communications Category M1 (**LTE MTC Cat M1**, also referred to as LTE-M) and Narrowband IoT (**NB-IoT**) [1], which provides ubiquitous and mobile connectivity to low cost and low power devices while improving both outdoor and indoor penetration coverage.

To ensure the success of the usage of **IoT** devices in the network, it is not only sufficient to provide efficient and low-cost connections, but also, it is necessary to be able to provide secure connectivity which is realized through strong authentication and encryption using the Subscriber Identity Module (**SIM**) card [2][3]. Unfortunately, the mentioned security mechanism is limited only towards the authentication, access control and encryption in the mobile network, since the exchanged messages are delivered to the **IoT** platform in clear text by the network.

In order to avoid this predicament, that means, to prevent the lack of security during the message exchange and to have adequate protection in the **IoT** platform overall, the platform must have its own encryption scheme as well the provisioning of authentication and access control mechanisms just as the **SIM** provides albeit it can be technic and economically challenging. To be able to overcome this limitation, leading to the purpose of this paper, is to introduce a cross layered Identity Federation, which offers single sign-on mechanisms and confidential/private access to the network to the **IoT** vertical sectors (e.g. health, transport, logistics, among others) using SIM authentication. This solution is developed within the scope of the **H2020 SCOTT** project [4] which is aiming at building trust in the Internet of Things.

This paper starts with a brief review of related work relevant for the background of this proposal, where within a review of identity management and its current standards is made. A brief description of the state-of-the-art on cellular **IoT** identity and access management is given, leading to our proposed solution with a usage scenario explained. It concludes with upcoming steps on our behalf as well as some suggestions for further works.

II. RELATED WORK

A. *SIM Authentication*

Even today, the usage of a SIM for authentication and access control in the mobile network has proven to be both affordable and trustable as a security measure, considering that there are several initiatives that want to extend its feasibility towards applications that connect with the Internet (browsing, e-mail, social networks, among others). For that purpose, the *Generic Bootstrapping Architecture* (**GBA**) [5][6] standard was introduced, specified by the 3rd Generation Partnership Project (**3GPP**), in which introduces a new element to the network – Bootstrapping Server Function (**BSF**) - responsible to retrieve an authentication vector from the Home Subscriber Server (**HSS**) and carrying out a mutual authentication of the mobile phone also known as *User Equipment* (**UE**). The **BSF** also

provides to a mobile Internet application, also known as a *Network Application Function* (**NAF**), an encryption key, K_{s_NAF}, for the session between itself and the **UE**. Although it fulfills the mentioned purpose, this solution is hindered by the demand of a **GBA** client in every mobile phone, and there is no incentive for Original Equipment Manufacturers (**OEM**) to implement it. To avoid this, the *Eureka Mobicome* project proposed some solutions regarding *SIM strong authentication*, in which strong authentication is provided from a regular browser to a mobile phone that carries a **SIM/USIM** [7][8].

ETSI did promote the usage of the **GBA** in their *Machine-to-Machine* (**M2M**) functional architecture [9], but despite its feasibility, it does not address **IoT** devices, in which the communication is done without the intervention of humans. So as of now, there isn't a comprehensive and flexible cellular **IoT** identity and access management system, enabling the inclusion of this type of device in the network with strong authentication and confidential/private access.

B. *Identitity Federation & Management*

In order to allow an **IoT** application or any digital service to use the network and its resources, the user must authenticate itself towards the service/application respective provider. Usually and nowadays, a set of user credentials (namely a user name and a password) allow the user to prove its authenticity towards set provider. However, the vastly increasing number of available services and the necessity of increased security obliged the user to make various different combinations with strict rules causing a burden to remember it all, not to mention that, for the service providers, it can also be an ordeal to ensure the security towards all threats out there, leading to an exhaustion of resources to address the matter.

To address this situation, so it can alleviate both users and service providers, mechanisms called identity federations were introduced, providing *Single Sign-On* (**SSO**) [10] solutions, which allow to simplify the registration and login processes for the user as well as reducing the costs for service providers while handling with their *Identity Management Systems* (**IDmS**).

1. OAuth 2.0

OAuth 2.0 [11] is an identity federation standard that allows the user to prove its authenticity towards a service by resorting to familiar third-party clients (e.g. Google, Facebook, among others), which means that is doesn't act as an *Identity Provider* (**IdP**) – notion first established by the Liberty Alliance Project [12], as an actor that permits the user and its accounts to authenticate itself towards a service provider – but solely as an Authentication Server, since it authenticates the user with its credentials from a third-party client, providing an authorization token to the application [13]. This mechanism ensures that a user will remain authenticated towards a service/application without going to the same sign-in/login process until the token is either revoked or expired.

2. OpenID Connect

The OpenID Connect [14] standard comes not only as an extension to OAuth for the authorization framework, but also goes one step further by offering **SSO** and identity provision mechanism on the internet. Nowadays, it is the most used solution even by popular identity providers such as *Google*, *Facebook*, among others, as well as mobile operators in their *Mobile Connect* solution, which provides a secure log-in mechanism for mobile phones. It is worth mentioning that it doesn't use identity federation but promotes the usage of the identity provider towards the service provider.

It is specified by a RESTful HTTP API and uses *JSON* as a data format with the purpose of enabling client applications to verify a user's identity due to the authentication process provided by an *OpenID Provider* and, at the end, such identity is given to the applications in an encoded *JSON web token* (**JWT**) known as *ID Token*.

As for its authorization flow, the service provider, hereby known as *Relying Party* (**RP**) initiates the process by redirecting the user to the *OpenID Provider* (**OP**) to access its identity, providing an open id value for each scope that it is requested. To ensure this, the provider will check if there is an active session, but in the case that there isn't, it will prompt the user for its consent to use its identity through the **RP**. Once allowed, an authorization code is provided, being stored for future utilizations, keeping in mind that the provided code will only sense once checked by the **OP** server and the **RP** can't use it on its own.

III. State of the Art

It is possible to consider the co-existing but independent behaviors and components an **IoT** has device in the network, mainly in the Network and the Application layers, as shown in Figure 1.

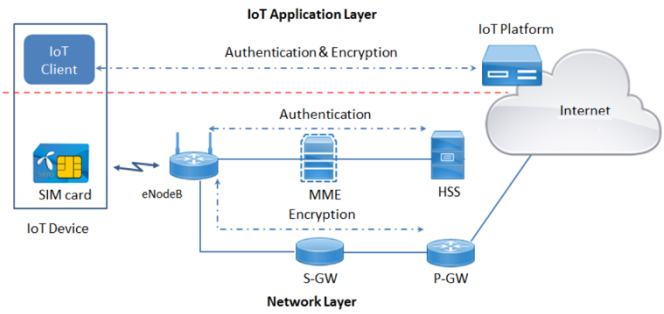

Fig. 1. Current Authentication and Access Control for **IoT** devices - Architecture

A. Network Layer

In the network, an **IoT** device can be identified by its *International Mobile Equipment Identity* (**IMEI**), the identity of the device, and the *International Mobile Subscriber Identity* (**IMSI**), that is associated to a SIM card, allowing a subscriber to have a unique identity. Upon power up, a mutual agreement process, using the *Authentication and Key Agreement* (**AKA**) [15] protocol, between the device and the network's Home Subscriber System (HSS) occurs and, once it is completed successfully, the device is allowed to be connected to the network and access its subscribed services. Also, to ensure the connection's integrity and confidentiality, encryption is applied in the radio access link using the cypher key that was exchanged during the authentication process and since it is a data connection, the *Packet Data Protocol* (**PDP**) is allocated that includes the mobile device's IP address.

B. Application Layer

After the network connection is established, communications through the application layer between an **IoT** device and an **IoT** platform can now occur. To prevent malicious attacks in these communications, authentication must be made between those two entities, as well as the usage of end-to-end encryption is recommended to ensure confidentiality. The device has a unique identity, IoT_{dev_ID}, solely recognizable by the **IoT** platform, which must be equipped with a suitable **IdMS** capable of performing strong authentication functions.

Alas, in order to satisfy these requirements, mobile operators would be obliged to make several and expensive investments towards their infrastructure, as well as hire expert technicians from the **IoT** provider to ensure support to the platform and the implemented management system. In order to alleviate this, the following proposal was conceived, which will be described in more detail.

IV. Proposed Solution

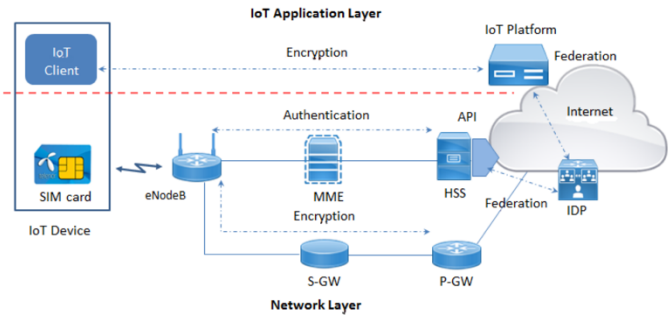

*Fig. 2. Proposed Solution for Authentication and Access Control for **IoT** devices - Architecture*

For our solution, we introduce a new entity in the network – *Identity Provider* (**IdP**) – whose mission is to bridge the gap between the Application and Network layers towards enabling a **SSO** mechanism for **IoT** devices. A *Subscriber Information Retrieval Application Programming Interface* (**SIR API**) is implemented on the **HSS**, whose interface is the standard Diameter-based *S6m* [16].

By establishing our own **IdP**, we are able to define which parameters or *scopes* shall constitute a user's identity in the network, i.e. an identity can be formed (through a set of key-value pair entries) by providing basic information such as a username and a password or it can become more complex when details about the user are provided to strengthen it. Once the user's identity is defined, the **IdMS** will store this information in a secure way, meaning that the system will have an independent database which can also be considered as a extension to the information stored at the network's **HSS**. This is where the **SIR API** will be applied since it has to prepare the user's identity, e.g. to gather the necessary data that the **HSS** can interpret .

Upon successful response from the **HSS** – ensuring that the **SIM** card used is in fact registered in the network - and registration in the **IdMS,** the user can be considered registered as well as the device he/she uses has become trustworthy. For further utilizations of the network, due to the **SSO** mechanism earlier mentioned, the authorization token that is generated using the identity provided by the system will be automatically renewed, providing a seamless experience, since the user does not have to provide his/her credentials any longer unless a logout, an expired session, a faulty operation or a blacklisted device has occurred.

To illustrate the procedure that our solution intents to provide, let us consider a small *"smart home"* **IoT** system composed by three cameras, one smoke detector and three contact sensors:

A. Configuration and Federation

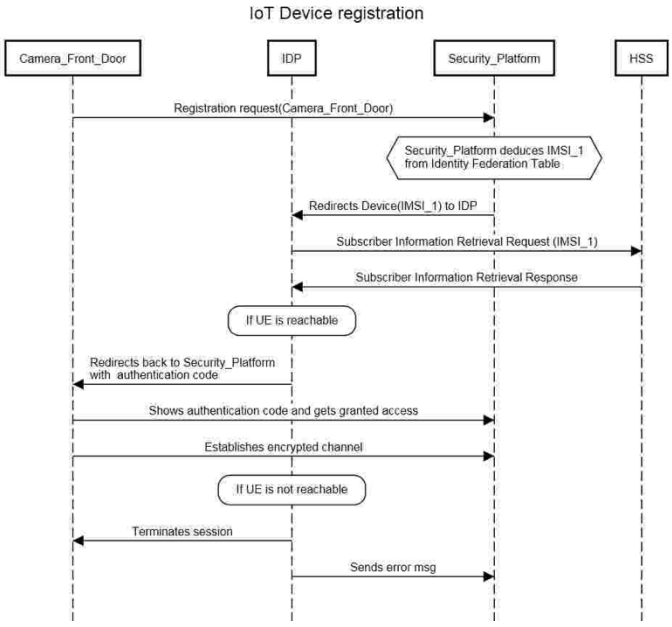

*Fig. 3. **IoT** Registration and Authentication example*

As mentioned earlier, this **IoT** system has seven devices, so it is necessary to acquire seven **SIM** cards - subscriptions - from a mobile operator. Each device gets a card assigned which contains an **IMSI**, in collusion with the its **IMEI** and an identity that corresponds its location. With these data, the **IoT** platform will carry out an identity federation so they are recognized in the network and the platform.

B. Authentication and Authorization

When all the devices have the **SIM** cards inserted and get properly installed at their location, the power is turned on, which will initiate the registration process. At the network layer, a mutual authentication process is carried out between the **SIM** card of the **IoT** device and the mobile network's **HSS**. Upon successful authentication, the **IoT** device is authorized to access the mobile network and it is granted with a data connection.

To perform its registration, each device sends a registration request to the security platform and gets redirected to the **IdP**. Afterwards, the **IdP** performs a **SIR** request at the **HSS** and receives the respective response, which contains information about whether the **UE** is registered to any serving node. If that is the case, the device has been successfully authenticated and the **IdP** can redirect the device with an authorization code back to the security platform, which will grant access in the network, in which an encrypted channel can be established between the device and the platform. In the case that the **UE** is not registered to any serving server one must consider that an anomaly has occurred, which could be an attack or simply a fault situation. To prevent any further damage the **IdP** will end the session with the device and send an error message to the security platform. Further actions should be carried out to find out what has happened to the device, mainly to verify if it was compromised.

V. IMPLEMENTATION

To test the feasibility of the proposed solution, a 4G *Long Term Evolution* (**LTE**) network was constructed at the Secure 5G4IoT Laboratory at Oslo Metropolitan University using open source software such as *OpenAirInterface*, developed by EURECOM [17].

A. Used Components

1. eNodeB

To provide a *eNodeB* (**eNB**), we utilize a generic PC running *Kali Linux* and *OpenAirInterface* connected to a *Universal Software Radio Peripheral* (**USRP**) N200, which software-defined and radios are designed and sold by Ettus Research [18].

2. Evolved Packet Core

To provide an Evolved Packet Core (**EPC**), we utilize a generic PC running *Ubuntu* and *OpenAirInterface*, which includes a **HSS** and a **SIR API**.

3. Identity Provider

The Identity Provider is established by using a generic PC running *Ubuntu Server* with *Gluu Server 3.1.2* [19] – an open source identity provider and management system as a server software bundle.

4. IoT Platform

The **IoT** platform is provided by using a generic PC running *Ubuntu Server* with *Gluu Server 3.1.2* and also a lightweight **M2M** open source server – Eclipse Leshan [20].

To simulate devices, due to lack of resources for the time being, we use Android devices with applications that use the *AppAuth* [21] Software Development Kit (SDK), as well as the full client provided by Gluu – *SuperGluu* [22], but also *Eclipse Leshan* clients. Since the developed work is still in an early stage, these clients can provide sufficient conclusions.

B. Tests

Our testing scenario which provides a solid proof of concept for the proposed solution and the beginning of further development at this stage of our research and development, consists of the following:

- Integration of the *GluuServer* in the mentioned network;
- Simple User Registration and Login;
- Managing authorization requests and tokens through the usage of the *AppAuth* library;

The integration of the server was ensured by fellow members of the Secure 5G4IoT Laboratory at Oslo Metropolitan University, where a firewall is established with proper forwarding rules, making it only reachable through the constructed network. The installation of the platform can be done following the documented procedures. It is recommended to have a dedicated computer for this server so there is no risk of conflicting with similar processes of other services, such as the Apache [24] project. Also, to ensure the minimum requirements of a well-scaled system, it is recommended to have a computer with a sizable amount of memory and storage. It is also relevant to mention that for our scenario, as it will be described later on, we have implemented native mobile applications to act as clients so that we're able to simulate the scenario in which a user interacts with its mobile device for various activities. This means that it is not recommended to use self-signed certificates in the server, since it does not comply with the standard practices of mobile application development when dealing with client/server communications [25].

Due to the flexibility of configuration that *GluuServer* provides, we are able to determine the definition of a user, i.e. its identity – as mentioned earlier; access methods - in this case, the definition of which applications/modules are allowed to act as clients in order to make requests to the system, as well as the types of authentication that can be used, such as basic authentication, *two factor authentication* (**2FA**), among others.

In our scenario, we've defined a user with the following scopes: username, first name, last name and email, and for this stage we have only considered basic authentication, so a user has to define a minimum eight characters password in order to authenticate itself in the system, but strong authentication methods will be used as we continue to develop our work. Following the premise of the mentioned network – a 5th generation cellular network, targeted for mobile and **IoT** devices, it made sense that one of the clients to be first considered was a native mobile application. With that, a simple Android app that uses the *AppAuth* library was used (and our own implementation is in the works), so that we can emulate the common scenario, in which a user, once confronted with an authentication process, whether it is a first time on a specific device or a first time usage which implies a registration in the system – to be issued with an identity by our **IdP** - or a further use of an already registered and authenticated device, the user can use its mobile device to verify the procedure.

In order to make this mobile application to be able to act as a client for our server, there is an initial configuration that needs to be done – mainly this is due to the fact that the application needs to obtain the discovery configuration of the server [26], which means it needs to know the location (in the network) of our **IdP**. With this configuration, we are now able to request authorization to access a certain service that exists in the network. Relying the identity to such services is under the scope of our future work.

The server must also be aware of the existence of this client. Even if the configuration above described is fully deployed, the server must issue a client id in order to make it trustworthy and be allowed to make requests. To do so, it is necessary to create a client entry of the server and configure it for our needs, such as define our own personalized authentication page, target application and most importantly the authentication flow that the client will use. It is also necessary to provide a *Unified Resource Locator* (**URL**) that allows to redirect the response from the server to the client. In this case the **URL** to use is what is commonly known as the *App Identifier*. At the end, these steps will determine all the necessary links between the parties involved in this authentication process. This configuration is not restricted to a user, but to a client which in this case is the mobile application. So, regardless of the number of the users that will use the network, the client (mobile application) needs only this unique configuration and can be deployed to multiple devices.

Once this client/server configuration is handled, which is recommended to do simultaneously to ensure that all parameters are linked between the parties, we are able to test this communication. At this stage, the user registration has handled separately, which means that it wasn't contemplated in this version of the app application, so a mobile browser was used to do so. *GluuServer* provides well defined endpoints for these procedures, such as:

- Register User URL: https://{idp-hostname}/identity/register
- Login User URL: https://{idp-hostname}/identity/login

In our version of the application we will allow both procedures to be handled in a more native format, but it is important to mention that the existence of this application is more a commodity for the user than an actual necessity, since the user can handle all these cases (registration, login, authentication/authorization) using solely the mobile browser.

Having a registered user, we are now able to test the app by making a request to the **IdMS** in order to authorize the usage of this new device and linking the user profile to it. Thus, the authorization flow initiates. The user will be redirected through the app to the server where it is asked to give is consent of the usage of his identity in order to deem the device trustworthy, allowing to use the network and its services.

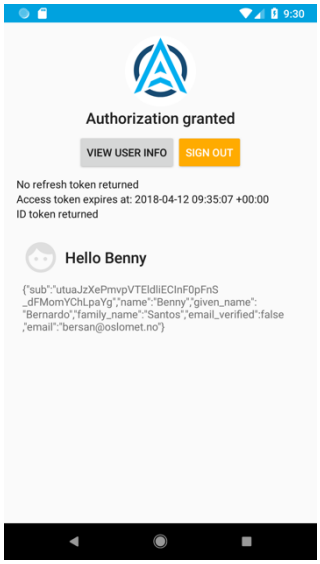

Fig. 4. AppAuth Application with authorization token obtained

In the Fig. 4 we have an example of a success use case in which a user requests authorization in the network by using its identity that was created in our **IdP**. As mentioned in II. and as it is perceived in the figure, the outcome of the authentication is an authorization token. In the app that we used, since we are in still in development stage, we see the content of the token as plaintext, but in a real scenario this information is not accessible and/or perceivable by the user at any point, in order to prevent inappropriate usage of such sensitive data.

VI. FUTURE WORK

As mentioned before, the proposed solution is in an early stage of implementation and, although it already shows promise as shown by provided scenario, more tests will be needed in order to be prepared for all kinds of scenarios. Also, those tests most consider real **IoT** devices in order to further study their behavior in the network, since considering their purpose - there is no need for an *always-on* connection, but once the device has data to provide to the platform, the exchange of data has to occur seamlessly, since the **SSO** mechanism is active and the device is recognized by the network and its services. Another field of work to consider is the design and implementation of a cross layer security solution aiming towards improving security of **IoT** systems by combining security measures deployed at both the mobile network layer and the **IoT** application layer.

VII. CONCLUSION

In this paper, an Identity Federation solution that allows the reuse of the SIM authentication carried out on the network layer for **IoT** applications and hence providing single sign on is presented. The proposed solution is proven to be feasible by making use of state-of-the-art open source software such as *OpenAirInterface*, *OpenID Connect* and *Eclipse IoT* [23]. With the proposed solution, **IoT** security will be enhanced while operational costs will be considerably reduced. The solution will pave the way for support of billions of **IoT** devices and application with the coming 5G mobile networks.

ACKNOWLEDGMENT

This paper is a result of the SCOTT project (www.scott-project.eu) which has received funding from the Electronic Component Systems for European Leadership Joint Undertaking under grant agreement No 737422. This Joint undertaking receives support from the European Union's Horizon 2020 research and innovation program and several countries such as Austria, Spain, Finland, Ireland, Sweden, Germany, Poland, Portugal, Netherlands, Belgium and Norway.

REFERENCES

[1] GSMA: 3GPP Low Power Wide Area Technologies: White paper 2.0, 2017

[2] 3GPP: TS 11.11 Specification of the Subscriber Identity Module - Mobile Equipment (SIM-ME) Interface, ver 8.14.0, 12-06-2007

[3] 3GPP: TS 31.102 Characteristics of the Universal Subscriber Identity Module (USIM) application ver 16-06-2017

[4] SCOTT: Secure Connected Trustable Things- https://scottproject.eu

[5] 3rd Generation Partnership Project: 3GPP TS 33.220 V8.2.0 (2007-12) Technical Specification Group Services and System Aspects; Generic Authentication Architecture (GAA) Generic bootstrapping architecture (Release 8)

[6] Timo Olkkonen: Generic Authentication Architecture, Helsinki University of Technology - http://www.tml.tkk.fi/Publications/C/22/papers/Olkkonen_final.pdf

[7] Do Van Thanh, Tore Jønvik, Do Van Thuan & Ivar Jørstad: Enhancing Internet service security using GSM SIM authentication, Proceedings of the IEEE Globecom2006 conference – ISBN 1-4244-0357-X – San Francisco, USA, Nov 27 - Dec 1, 2006

[8] Do van Thanh, Tore Jønvik, Boning Feng, Do van Thuan & Ivar Jørstad: Simple Strong Authentication for Internet Applications using mobile phones, Proceedings of IEEE Global Communications Conference (IEEE GLOBECOM 2008), ISBN 978-1-4244-2324-8, New Orleans, LA, USA, Nov 30 – Dec 4, 2008

[9] ETSI: TS 102 921 Machine-to-Machine communications (M2M); mIa, dIa and mId interfaces, V2.1.1 (2013-12)

[10] Telektronikk 3/4 2007: Identity Management – Guest Editorial Do van Thanh – ISSN 0085-7130 - https://www.telenor.com/wp-content/uploads/2012/05/T07_3-4.pdf

[11] IETF Request for Comments: 6749: The OAuth 2.0 Authorization Framework, October 2012

[12] Liberty Alliance: ID-FF Architecture Overview – vers. 1.2-errata-v1.0.

[13] Anicas Mitchell: An Introduction to OAuth 2, posted Jul 21, 2014 - https://www.digitalocean.com/community/tutorials/an-introduction-to-oauth-2

[14] OpenID Connect: http://openid.net/connect/

[15] ETSI: TS 133 102 v3.6.0 (2000-10) Universal Mobile Telecommunications System (UMTS); 3G Security; Security Architecture

[16] 3GPP: TS 29.336 V15.0.0 (2017-09) Technical Specification 3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; Home Subscriber Server (HSS) diameter interfaces for interworking with packet data networks and applications (Release 15)

[17] The OpenAirInterfaceTM Software Alliance (OSA) http://www.openairinterface.org/

[18] Ettus Research: https://www.ettus.com/

[19] Gluu Server: https://www.gluu.org/

[20] Eclipse Leshan: https://eclipse.org/leshan/

[21] AppAuth: https://appauth.io/

[22] Super Gluu: https://super.gluu.org/

[23] Open Source for IoT: https://iot.eclipse.org/

[24] Apache: https://httpd.apache.org/

[25] Android – Security with HTTPS and SSL: https://developer.android.com/training/articles/security-ssl.html#SelfSigned

[25] Open ID Connect Discovery : http://openid.net/specs/openid-connect-discovery-1_0-21.html