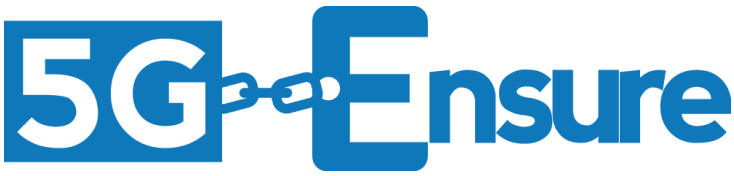

Deliverable D3.8
5G-PPP Security Enablers Documentation (v2.0)
Enabler Internet of Things

Project name	5G Enablers for Network and System Security and Resilience	
Short name	5G-ENSURE	
Grant agreement	671562	
Call	H2020-ICT-2014-2	
Delivery date	31.08.2017	
Dissemination Level:	Public	
Lead beneficiary	NEC	Felix Klaedtke, felix.klaedtke@neclab.eu
Authors	SICS: Markus Ahlström, Rosario Giustolisi, Simon Holmberg, Martin Svensson	

This part will be removed before the final edition and publication

Document Version	Date	Change(s)	Author(s)
0.1	02.06.2017	Created template	Felix Klaedtke
0.2	8.8.2017	Reviewed. Corrected typos	Aleksi Dahl

Foreword

5G-ENSURE belongs to the first group of EU-funded projects which collaboratively develop 5G under the umbrella of the 5G Infrastructure Public Private Partnership (5G-PPP) in the Horizon 2020 Programme. The overall goal of 5G-ENSURE is to deliver strategic impact across technology and business enablement, standardization and vision for a secure, resilient and viable 5G network. The project covers research and innovation - from technical solutions (5G security architecture and testbed with 5G security enablers) to market validation and stakeholder's engagement - spanning various application domains.

This document is a part of D3.8 and is an updated version of deliverable D3.4 5G-PPP Security Enablers Documentation (v1.0). This document concerns documentation of the Release 2 implementation of the Internet of Things Enabler feature Group-based authentication.

Disclaimer

Copyright notice

Contents

1 Introduction

The number of machine-type communications (MTC) devices is predicted to increase enormously with 5G. This will lead to an increased signaling that may congestion and increase the network access latency. Sport events is one example use case where the consequences outlined above may happen. During sport events, a large number of sensors and tracking devices are used to increase the user experience [1]. By grouping devices on the basis of features such as ownership or location, the signaling of the Authentication and Key Agreement (AKA) can be reduced.

The Internet of things enabler has four features, release 2 includes the "Group-based authentication" feature. Section 2.2 of Deliverable 3.6 [1] describes the group-based AKA enabler feature of which a prototype implementation has been developed for release 1 and 2. Basically, the enabler consists of a new AKA protocol that decreases both network access latency and data traffic between the MME and the HSS. The new protocol allows MTC devices to perform AKA procedures without the need that the MME communicates with the HSS. This is achieved by using inverted hash trees containing authentication information. When the first MTC device attaches, Case A is performed: the MTC device is authenticated as in EPS AKA but the HSS includes sub-root nodes in the group trees along with the authentication vector (AV). When the remaining MTC devices in the group perform Case B of the protocol, individual authentication parameters are derived by the MME from the sub-root nodes so that the HSS does not have to be contacted, which is beneficial during roaming. In release 2 multiple UE instances have been added and sequence numbers are introduced in the protocol to enable re-authentication of UE. For a complete description of the protocol see the Group-based authentication article by R. Giustolisi [2] .

The enabler feature is implemented as an extension of an open-source EPS project called OpenAirInterface (OAI). The project includes two major software, "Openair-cn" and "OpenAirInterface5G". General information about OAI can be found at their website [3].

2 Installation and Administration Guide

Release 2 of the enabler feature runs on two hosts, see Figure 1: Illustration of the considered OAI deployment.for an example deployment of the feature. The first host runs an OAI UE and an OAI eNodeB software. The second host runs SPGW, MME and HSS software.

The installation process starts by installing a Debian package called *group-auth-oaisim_2.0.deb* on the first host and then installing the Debian packages *group-auth-asn1c_2.0_all.deb*, *group-auth-freediameter_2.0_all.deb*, *group-auth-gtpu_2.0_all.deb*, *group-auth-epc_2.0_all.deb* on the second host.

After the Debian packages are installed then the configuration of the network interfaces, OAI and MySQL should be performed through two Ansible scripts. There is one ansible script for each host. Regarding the group-based authentication configuration for each UE (such as IMSI, GID, long-term key, etc.), it is done in the file *ue_eurecom_test_sfr.conf* in the first host and the corresponding configuration for the EPC is done in a MySQL database called OAI_DB on the second host. See Figure 2: Illustration of example network architecture for the enabler on two hosts.for an example network architecture of the implementation and view Enabler Installation and Enabler Configuration for detailed instructions.

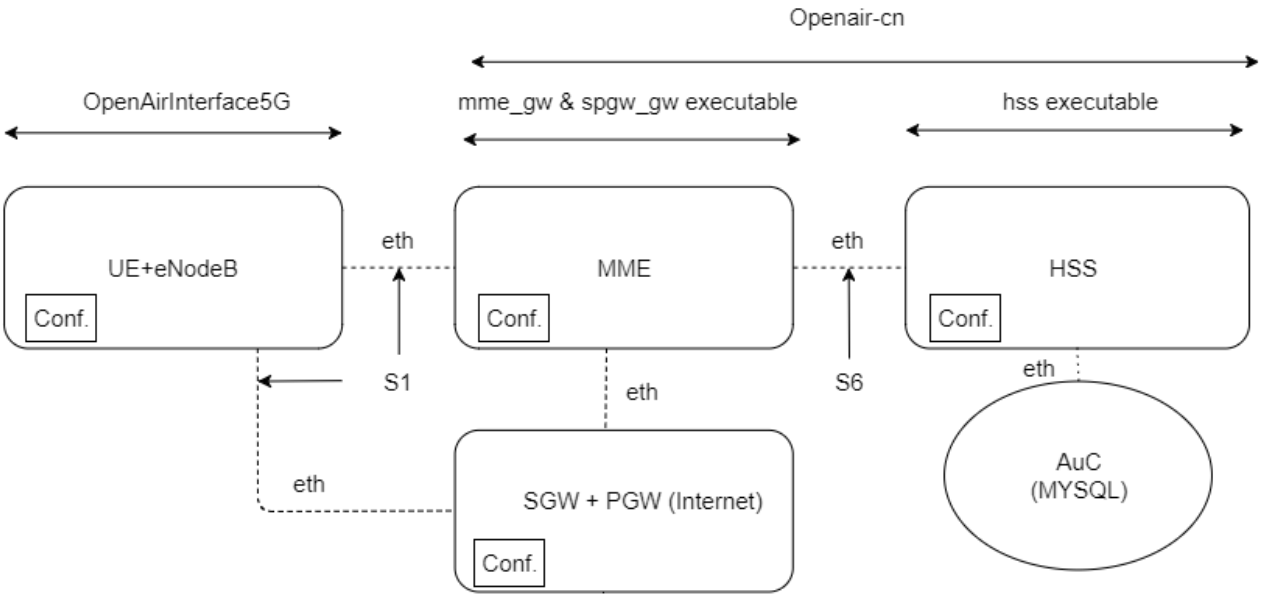

Figure 1: Illustration of the considered OAI deployment.

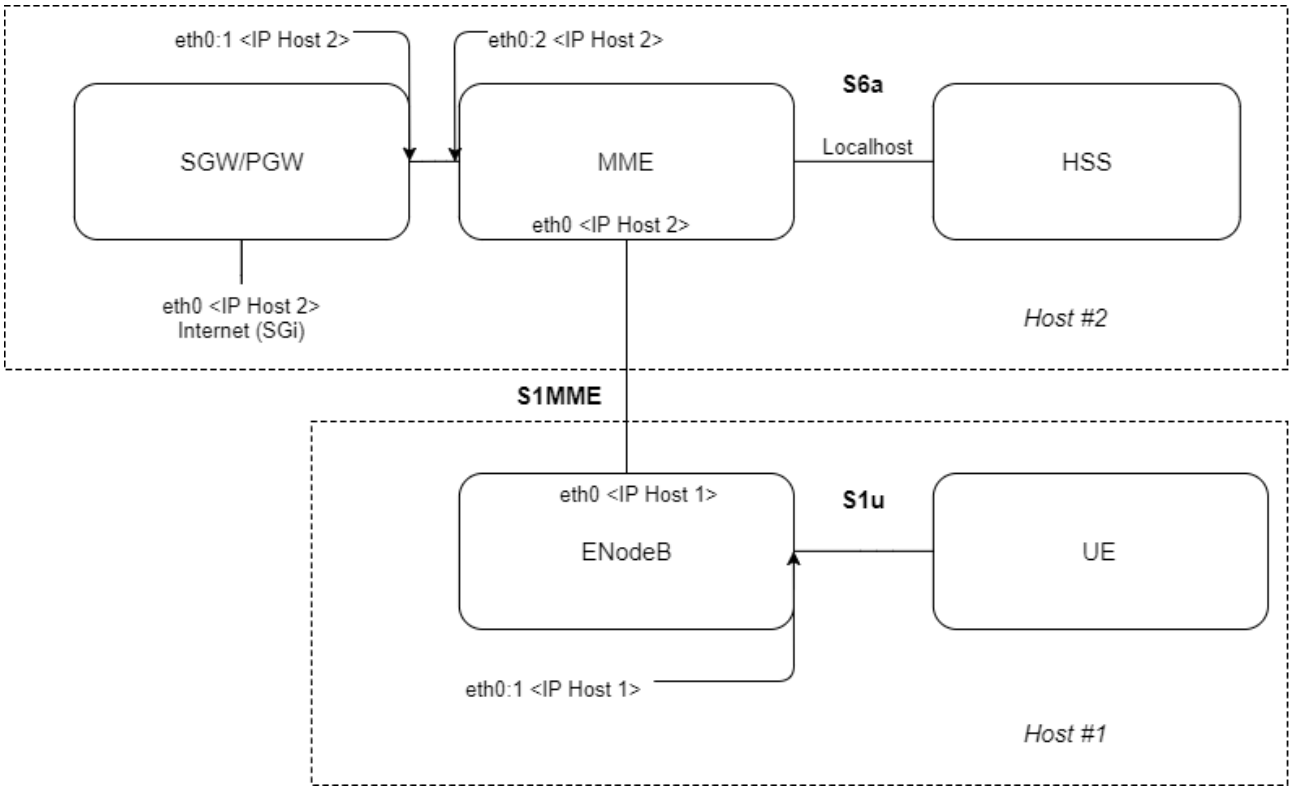

Figure 2: Illustration of example network architecture for the enabler on two hosts.

2.1 System Requirements

The enabler is built on OAI and has been tested on the operating system Ubuntu 14.04 LTS (Trusty Tahr) using 4.7.1 kernel on Intel x86 64 bits platforms. Consequently, we recommend to use the same Ubuntu release and kernel for the enabler. The enabler requires GTPU kernel module to be installed.

The enabler was developed and tested using the Generation 3 Intel Core i7 processor with 4GB RAM. But the OAI software has also been tested on the following processor families: Generation 3 Intel Core i5, Generation 2 Intel Xeon and Intel Atom Rangely; hence, the enabler should also work with these processors families. Note that the current OpenAirInterface software requires Intel architecture based PCs.

2.2 Enabler Installation

The files included in the enabler installation consists of five Debian packages and two ansible scripts.

2.2.1 UE/eNodeB Installation on Host #1

The installation of the UE/eNodeB is done using a standard Debian package:

$: sudo dkpg -i *group-auth-oaisim_2.0.deb*

Any missing dependencies can be installed using:

$: sudo apt-get -f install

2.2.2 MME/SPGW/HSS Installation on Host #2

The installation of the MME, SPGW and HSS is done by installing MySQL and then using standard Debian packages. The order of installation of the Debian packages which follows is recommended:

$: sudo apt-get install mysql-server mysql-client

$: sudo dpkg -i *group-auth-asn1c_2.0_all.deb*

$: sudo dpkg -i *group-auth-freediameter_2.0_all.deb*

$: sudo dpkg -i *group-auth-gtpu_2.0_all.deb*

$: sudo dpkg -i *group-auth-epc_2.0_all.deb*

Any missing dependencies after any step above can be installed using:

$: sudo apt-get -f install

2.3 Enabler Configuration

The main configuration is done automatically using two Ansible [4] scripts, one for each host. They should be performed after enabler installation. Configuration of authentication parameters for each UE is done on the first host (UE/eNodeB) in a configuration file called *ue_eurecom_test_sfr.conf* and for the second host in a database called OAI_DB in MySQL.

2.3.1 UE/eNodeB configuration on Host #1

The ansible script used on the first host is called *OAISIMconf.yml* in which the parameters that must be set in that file are described in Table 1: Parameters of OAISIMconf.yml to be set.

Table 1: Parameters of OAISIMconf.yml to be set

Parameter name	Explanation
mme_vm_ip	The IP address of the second host
sim_vm_interface	The interface used by the first host
sim_vm_ip	The IP address of the first host

Then the script should be executed, it sets the correct values in various configuration files for OAI:

$: ansible-playbook *OAISIMconf.yml*

The file *ue_eurecom_test_sfr.conf* is in folder /usr/local/oai/NAS/TOOLS/ and can be modified to change or add UE parameters including Group based authentication parameters. Note that changing these parameters also require that the database on the second host must also be configured with the same changes or additions. It is assumed that all EPS related UE parameters are known to the reader, however for the Group based authentication parameters, they are explained in the D3.6 Open Specification document [5]. See an excerpt from the configuration file below.

```
# List of known PLMNS
PLMN: {
    PLMN0: {
           FULLNAME="Test network";
           SHORTNAME="OAI4G";
           MNC="01";
           MCC="001";
};

AKA:
{
          # Group-based AKA boolean. "1" if group-based AKA should be used, "0" if EPS-AKA should be used.
          GAKA="1";
}

UE0:
{
    USER: {
        IMEI="35609304079211";
        MANUFACTURER="EURECOM";
        MODEL="LTE Android PC";
        PIN="0000";
        GAKA_GID="208935555555555";
        GAKA_PATH="22";
        GAKA_OBFUSCATED_VALUE="D075FA8CF2D99873DBDEA4B923088522";
    };

    SIM: {
        MSIN="0100001111";
        USIM_API_K="8bAF473F2F8FD09487CCCBD7097C6862";
        OPC="C42449363BBAD02B66D16BC975D77CC1";
        MSISDN="33611123456";
    };

    # Home PLMN Selector with Access Technology
    HPLMN= "20893";

    # User controlled PLMN Selector with Access Technology
    UCPLMN_LIST = ();

    # Operator PLMN List
    OPLMN_LIST = ("00101", "20810", "20811", "20813", "20893", "310280", "310028");

    # Operator controlled PLMN Selector with Access Technology
    OCPLMN_LIST = ("22210", "21401", "21406", "26202", "26204");

    # Forbidden plmns
    FPLMN_LIST = ();

    # List of Equivalent HPLMNs
#TODO: UE does not connect if set, to be fixed in the UE
#    EHPLMN_LIST= ("20811", "20813");
    EHPLMN_LIST= ();
};
UE2: . . . add here
```

After eventual changes are made the following script must be run in the same folder to update stored UE information files in a bin folder:

$: ./conf2uedata *ue_eurecom_test_sfr.conf ../../targets/bin*

Note that this file is preset with correct settings for UEs and it is recommended to be kept as is.

2.3.2 MME/SPGW/HSS Configuration on Host #2

The ansible script used on the second host is called *EPCconf.yml* in which the parameters that must be set in that file are described in Table 2 : Parameters of EPCconf.yml to be set

Table 2 : Parameters of EPCconf.yml to be set

Parameter name	Explanation
hss_vm_fqdn	The FQDN of the HSS
mysql_user_name	Your chosen MySQL user-name
mysql_user_password	Your chosen MySQL password
mme_vm_fqdn	The FQDN of the MME
mme_vm_realm	The realm of the MME
hss_vm_realm	The realm of the HSS
hss_vm_hostname	The hostname of the HSS
mme_vm_interface	The interface used by the MME (host #2)
mme_vm_ip	The IP address used by the MME (host #2)

Then the script should be executed, it sets the correct values in various configuration files for OAI:

$: ansible-playbook *EPCconf.yml*

A new MySQL database called OAI_DB should now exist on the second host. It contains (among others) the tables *groups*, *users*, and *paths* which can be altered to change or add UE information and Group based authentication parameters. It is assumed that all EPS related UE parameters are known to the reader, however for the Group based authentication parameters, they are explained in the D3.6 Open Specification document [5]. It is recommended to keep the pre-set configuration of the database, however if any change is made remember to make corresponding changes to the first host UE/eNodeB configuration file *ue_eurecom_test_sfr.conf*.

2.4 Troubleshooting

Only the communications between the UE and the core network that concerns the AKA protocol are considered. Hence, any issues or errors concerning other aspects of the OAI such as radio simulation errors or lower parts of the stack can be ignored. In the scope of release 2, a real error occurs if the UE and the CN fail to authenticate each other and consequently if they derive different session keys.

During the installation of Openair-cn, MySQL will sometimes not start after MySQL has been installed. If this happens, open a new terminal and start it manually by entering:

$: sudo /etc/init.d/mysql start

A lot of useful information about the software and some known bugs can be found in the OAI repository wiki [6].

3 User and Programmer Guide

3.1 User Guide

This section can only be performed after enabler installation and enabler configuration. The MME/SPGW/HSS must be running on the second host before starting the UE and eNodeB on the first host. You can exit any of the scripts them by using the keyboard shortcut `Control-C`.

3.1.1 Running MME/SPGW/HSS on Host #2

The MME, SPGW and HSS require three terminals to be run. Navigate to the folder */usr/local/etc/oai/SCRIPTS* in all three terminals. First, to start the MME, execute the following in the first terminal:

$: ./run_mme -i

The MME will wait for SPGW and HSS connection before continuing. Advance by executing the following in the second terminal:

$: ./run_spgw

Lastly, start the HSS in the third terminal by executing the following:

$: ./run_hss

The excepted result of this should be an OPEN_STATE printed by the the ./run_hss command and ./run_mme command.

3.1.2 Running UE/eNodeB on Host #1

The UE/eNodeB require one terminal to be run. Navigate to the folder */usr/local/etc/oai/cmake_targets/tools*. Execute the following command:

$: sudo -E ./run_enb_ue_virt_s1

3.2 Programmer Guide

The enabler is not programmable.

4 Unit Tests

4.1 Information about Tests

These tests are made to assert that the UE/eNodeB and MySQL database, MME, SPGW and HSS are operational. No interconnection tests are made between the hosts since such tests would be in the domain of security evaluation tests.

4.2 Unit Test 1: MME, SPGW, HSS and MySQL operationally

On the second host (MME, HSS, SPGW) open a terminal. Execute the following:

1. $: mysql -u root -p<your MySQL password>
2. use oai_db;
3. show tables;

Expected result:

```
+------------------------+
| Tables_in_oai_db |
+------------------------+
| apn                    |
| mmeidentity          |
| pdn                    |
| pgw                   |
| terminal-info         |
| users                  |
| paths                  |
| groups                 |
+------------------------+
```

Next open four terminals on the host. On one, execute:

$: sudo tcpdump -i any -s 0 -w my.pcap

On the second execute:

$: /usr/local/etc/oai/SCRIPTS/run_hss

"Initializing s6a layer: DONE" Should be printed.

On the third terminal execute:

$: /usr/local/etc/oai/SCRIPTS/run_mme -i

“'STATE_WAITCEA' -> 'STATE_OPEN'” should be printed eventually (after a lot of other printouts). The same should be printed on the second terminal.

On the fourth terminal execute:

$: /usr/local/etc/oai/SCRIPTS/run_spgw

“Initializing GTPU1U interface: DONE” and “Initializing SPGW-APP task interface: DONE” should be printed.

In the first terminal execute:

$: tshark -n -r my.pcap -d “tcp.port=3870, diameter” | tee Diamter.log”

Expected result: Diameter.log should be produced and contain records of Diameter “Capabilities-Exchange” request and answer and “Device-Watchdog” request and answer”. If this is true and the previous steps are true, then the test is successful.

Now stop the programs running on the three terminals (Control-c).

4.3 Unit Test 2: UE/eNodeB operationality

On the first host traverse to the folder /usr/locat/etc/oai/cmake_targets/tools. Then execute:

$: sudo -E ./run_enb_ue_virt_s1

Expected result: The following text should be printed eventually: “[ENB_APP][W] 1 eNB is not associated with a MME, retrying registration in 10 seconds …”

5 Abbreviations

5G-PPP	5G Infrastructure Public Private Partnership
OAI	OpenAirInterface
UE	User Equipment
eNodeB	Evolved Node B
MME	Mobility Management Entity
HSS	Home Subscriber Server
EPC	Evolved Packet Core
SPGW	Serving Gateway/PDN Gateway
AKA	Authentication and Key Agreement

6 References

[1] "2017., 5G-ENSURE Consortium. Deliverable D3.6: 5G-PPP security enablers open specifications (v2.0). Available online:http://www.5gensure.eu/sites/default/files/5G-ENSURE_D3.6%205G-PPP%20security%20enablers%20open%20specifications%20%28v2.0%29.pdf".

[2] R. e. a. Giustolisi, ""A Secure Group-Based AKA Protocol for Machine-Type Communications."," *International Conference on Information Security and Cryptology,* Vols. Springer, Cham, 2016.

[3] "OpenAirInterface Website: http://www.openairinterface.org/ 2017".

[4] "Ansible: http://docs.ansible.com/".

[5] "Deliverable D3.6 Open Specifications v2.0," [Online]. Available: http://www.5gensure.eu/sites/default/files/5G-ENSURE_D3.6%205G-PPP%20security%20enablers%20open%20specifications%20%28v2.0%29.pdf.

[6] "OAI wiki," [Online]. Available: https://gitlab.eurecom.fr/oai/openairinterface5g/wikis/home.