

**Software Design Document**  
for  
**RADIUS Plugin for Strengthening Wireless**  
**Authentication of Devices**  
Government Engineering College, Thrissur

Bony Tom  
Chippy Sasankan

November 24, 2020

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Purpose . . . . .	2
1.2	Scope . . . . .	2
1.3	Definitions, Acronyms and Abbreviations . . . . .	2
1.3.1	Definition . . . . .	2
1.4	Abbreviations . . . . .	2
<b>2</b>	<b>References</b>	<b>2</b>
<b>3</b>	<b>Architectural Description</b>	<b>3</b>
<b>4</b>	<b>Decomposition Description</b>	<b>4</b>
4.0.1	Component Decomposition . . . . .	4
<b>5</b>	<b>Dependency Description</b>	<b>5</b>
5.1	Intermodule Dependency . . . . .	5
<b>6</b>	<b>User Interface Description</b>	<b>6</b>
6.1	Registration User Interface . . . . .	6
6.2	Client User Interface . . . . .	6
<b>7</b>	<b>Detailed Design</b>	<b>8</b>
7.1	Module Detailed Design . . . . .	8
7.1.1	Registration . . . . .	8
7.1.2	Plugin . . . . .	8

# 1 Introduction

## 1.1 Purpose

The purpose of this document is to delineate the design of our project titled - RADIUS Plugin for Strengthening Wireless Authentication of devices in different Design Viewpoints..

## 1.2 Scope

Our freeRADIUS Plugin aims to strengthen authentication of users connecting to a network, especially over a wireless network, especially from the current threats of MAC Address Sniffing and Spoofing

## 1.3 Definitions, Acronyms and Abbreviations

### 1.3.1 Definition

- **Authentication** :- Authentication is the act of confirming the truth of an attribute of a datum or entity. In Computer Networks, it is verifying that a client is what it claims to be.
- **Supplicant**:- In IEEE 802.1x Authentication protocol Scenario, Supplicant is the devices requesting access to the network resources. It is the device that supplies information to the RADIUS Server.
- **Authenticator**:- Authenticator in IEEE 802.1x Scenario is the Access Point/Router that communicates with the Authentication Server(RADIUS, etc) to authenticate Supplicants requesting access to the network.
- **Client** :- Clients in the scope of this document are devices requesting access to the network. In the Scope of this document, Clients and Supplicants are used interchangeably.

## 1.4 Abbreviations

- **RADIUS** :- Remote Authentication Dial In User Service
- **MAC** :- Media Access Control ( or Ethernet Hardware Address )
- **EAP** :- Extensible Authentication Protocol
- **EAPoL** :- Extensible Authentication Protocol over LAN
- **NAS** :- Network Access Server
- **PAE** :- Port Access Entity

# 2 References

- IEEE SDD Standard -SDD-ieee-1016-2009.pdf in local folder
- RADIUS RFC - <http://tools.ietf.org/html/RFC2865>
- EAP RFC - <http://tools.ietf.org/html/rfc3748>

### 3 Architectural Description

The Architecture comprises of the Static and Dynamic Aspects of the System. It gives a view of the entire system highlighting the important features and ignoring unnecessary details. The System is mainly a Authentication Server - Authenticator - Supplicant Model where RADIUS Messages are passed between the Authentication Server (RADIUS Server) and the Authenticator (NAS or Wireless Router, etc), and EAPoL Messages are passed between the Supplicant (Client Component of the System) and the Authenticator.

The Actual Messages passed between the different components of the Architecture are shown in Fig 2.

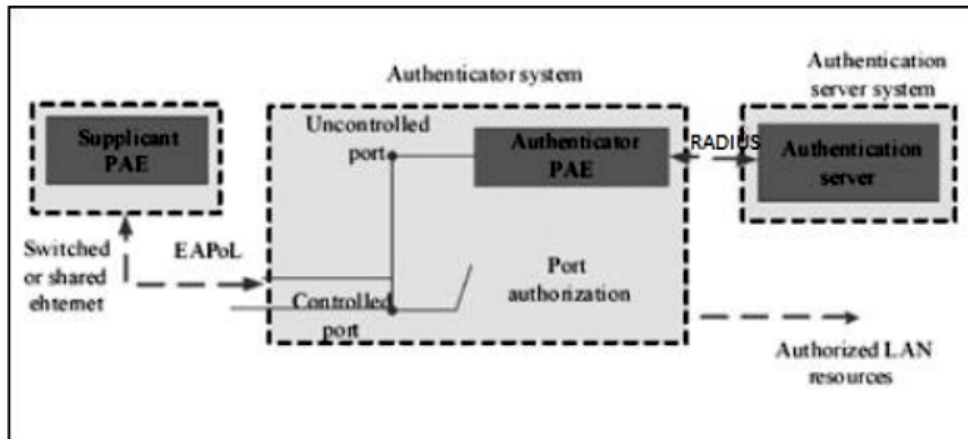


Figure 1: Basic IEEE 802.1x Authentication Architecture

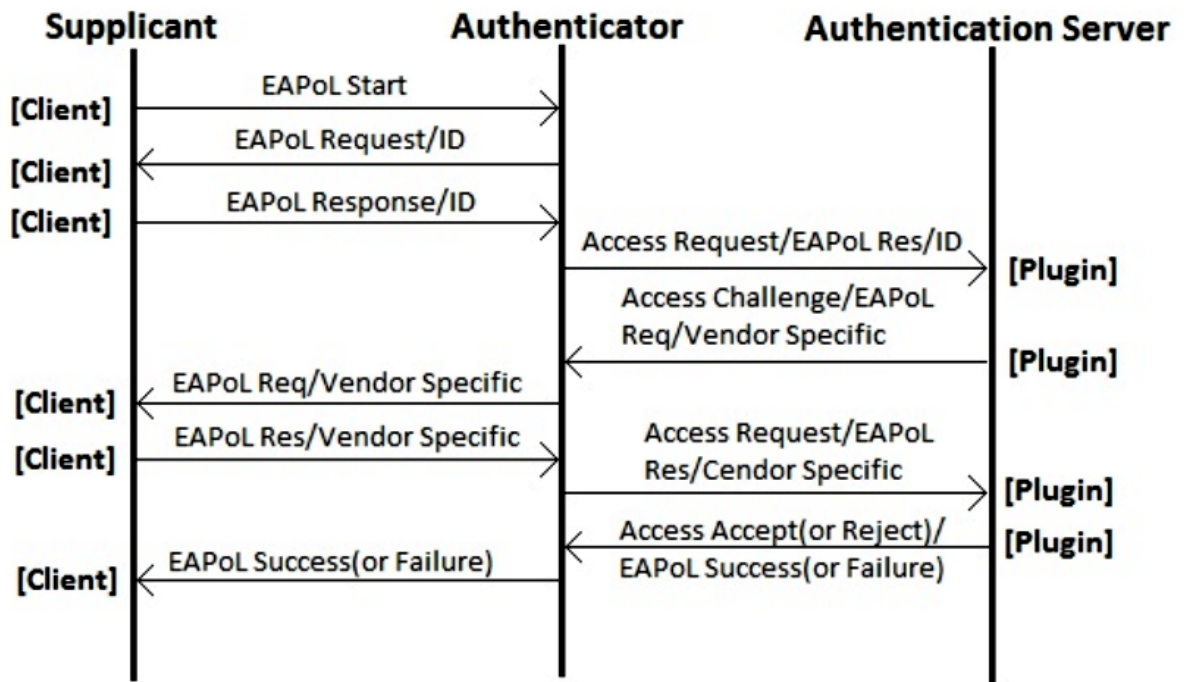


Figure 2: Basic IEEE 802.1x Authentication Architecture

## 4 Decomposition Description

The System is divided into 3 modules based on the functionality of the system, that is it is divided into modules for each of the functionality the system would provide. The Modules are :-

- **Registration** - This Module consists of the Registration user interface. It would receive request from the plugin containing the generated key, which the Registration authority should compare with the key generated in the client, by the Client Module, and should respond with either Register or Dont Register.
- **Plugin** - This Module would be called when the Access-Request packet contains a MAC Address for which a key is present in the database. The purpose of this Module would be to acquire the data for generation of key and compare it with stored key and reply with either Access-Accept (if keys are same) or Access-Reject (if keys are different) packet. And if no key is stored in RADIUS database for a particular MAC Address, it should send the generated key to Registration User Interface and Store they key and combination in the Database only if it receives a Register Response from the interface
- **Client Module** - This Module runs in the supplicant and communicates with the Authenticator using EAP, and acquires different attributes of the supplicant Device (Client Device) when requested and responds with those attributes in EAP Messages. The Attributes that should be supported by this module include :-
  - OS Name and OS Version.
  - CPU ID.
  - BIOS ID.
  - Motherboard ID.

It should also generate key using combination of all (as would be generated by the Plugin) and show it, when requested using a Show Key button on its user. Interface

### 4.0.1 Component Decomposition

The System is divided into 3 Components running on different Computers, one on the Supplicant Computer, and the other on the Authentication Server

- The Component running on Supplicant would receive EAPoL requests from Authenticator and respond to the requests accordingly.
  - When Requested for OS, Client should reply with OS Name + OS Version.
  - When Requested for CPU ID, Client should reply with the CPU ID
  - When Requested for BIOS ID, Client should reply with the BIOS ID
  - When Requested for Motherboard ID, Client should reply with Motherboard ID.

It should also generate key using combination of all (as would be generated by the Plugin) and show it, when requested using a Show Key button on its user.

- The Component running on the Authentication Server (RADIUS Server), would be the plugin. It would be responsible for key generation and authentication of supplicants.
- Another Component would be running on the Authentication Server (RADIUS Server), that would be the Registration User Interface. It would be responsible for receiving Generated key (of combination of all attributes) and showing it for comparison with key generated in the device, and responding with either Register or Dont Register as chosen by the Registration Authority.

## 5 Dependency Description

### 5.1 Intermodule Dependency

The Dependencies between the modules is expressed as the DFD diagram shown below :



Figure 3: Data Flow Diagram Lvl 0

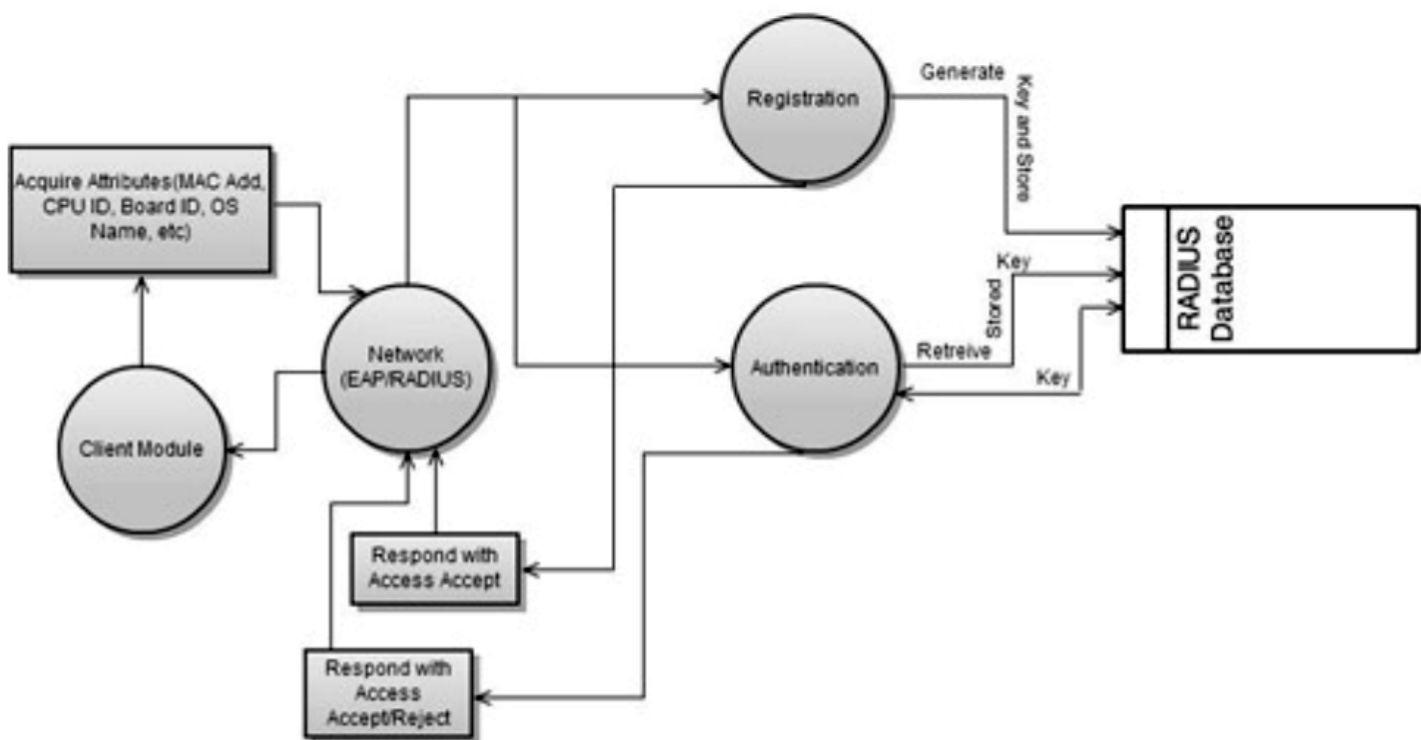


Figure 4: Data Flow Diagram Lvl 1

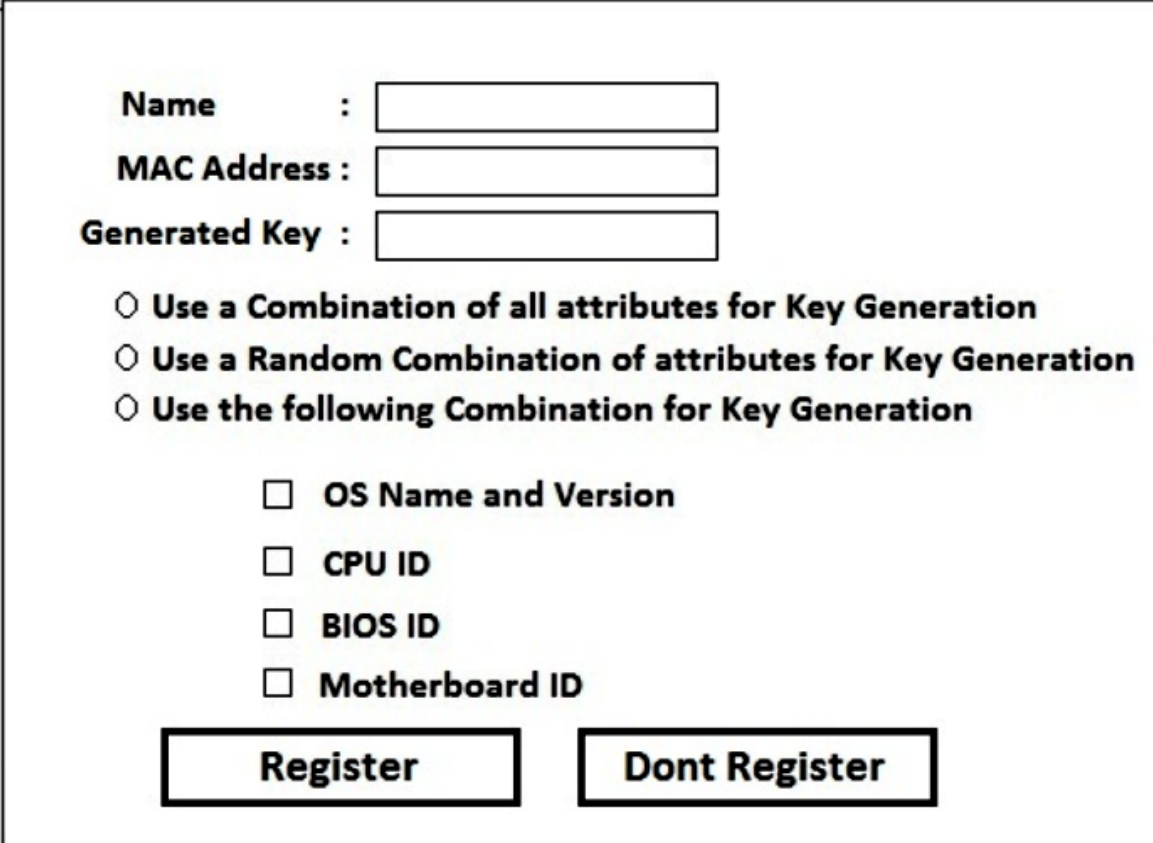
## 6 User Interface Description

### 6.1 Registration User Interface

There would be a Registration user Interface accessible to the Registration Authority. The interface would contain option to show the MAC Address of the Device being registered, show the Generated key it recieved from the Plugin and Enter the Name of Person registering it. And options to select what Device informations the system should acquire for Key generation and Authentication. The Different options can be :-

- OS Name and OS Version.
- CPU ID.
- BIOS ID.
- Motherboard ID.

The interface would also contain options to use random combination, or use all.



The form is enclosed in a rectangular border. It contains the following elements from top to bottom:

- A label **Name** followed by a colon and a text input field.
- A label **MAC Address** followed by a colon and a text input field.
- A label **Generated Key** followed by a colon and a text input field.
- Three radio button options:
  - ☐ **Use a Combination of all attributes for Key Generation**
  - ☐ **Use a Random Combination of attributes for Key Generation**
  - ☐ **Use the following Combination for Key Generation**
- Four checkbox options:
  - ☐ **OS Name and Version**
  - ☐ **CPU ID**
  - ☐ **BIOS ID**
  - ☐ **Motherboard ID**
- Two buttons at the bottom: **Register** and **Dont Register**.

Figure 5: Registration User Interface

### 6.2 Client User Interface

The Client User Interface would contain option to Generate Key. The Generated key would be shown in the same window above the button. The Client side user interface would look like :-

The diagram illustrates a client user interface within a rectangular frame. It features a label **Generated Key :** followed by a rectangular input field. Below the input field is a button labeled **Generate Key**.

Figure 6: Client User Interface



## 7 Detailed Design

### 7.1 Module Detailed Design

#### 1. Authentication

- (a) On Startup, Client Module sends EAP Start message into the network, On Receipt of EAP Request/Identity Message, it should reply with EAP Response/Identity, MAC Address should be used as the identity. It should then keep listening for data in the network.
- (b) On Receipt of EAP Request/Vendor Specific Packet, it should acquire the required information from the device (depending of the EAP Request packet it received) and respond with that information in EAP Response/Vendor Specific Packet.

#### 2. Key Generation

- (a) When Generate Key button is pressed, the System should acquire the all the information stated in Client Module in Section 4, and use AES algorithm with the MAC Address as key and the information concatenated is a pre-fixed sequence as plaintext to generate the key and Display it on Screen.

#### 7.1.1 Registration

1. If the Registration Module receives a MAC Address from the Plugin, it should display it to the screen and request confirmation that the above MAC Address is for Registration, if the Registration authority clicks -Register, it should return Continue Registration message (Can be encoded) to the plugin.
2. If the Registration module receives Generated Key from the Plugin, it should display it to the screen along with MAC Address and request confirmation that the above MAC Address is for Registration and the Key Generated is correct, if the Registration authority clicks -Register, it should return Confirm Registration message (Can be encoded) to the plugin.

#### 7.1.2 Plugin

1. The Plugin starts in State 0, for all MAC Addresses.
2. When the Plugin receives an Access-Request Packet from authenticator, containing a MAC Address in State 0 with the MAC Address and generated key both present in the Database, it should produce an EAP Request/Vendor Specific Packet with the MAC Address in the Vendor Specific Data part of the packet and encapsulate it in Access-Challenge Packet and send to the Authenticator. It should save the state for the corresponding MAC Address as State 1. If the MAC Address is not contained in the Database, then the Plugin should go to State 2, and pass the MAC Address to the Registration Module installed on the same computer.
3. When the Plugin receives an Access-Request Packet from authenticator, containing a MAC Address in State 1 with the MAC Address and generated key both present in the Database, it should parse the EAP Response/Vendor Specific Packet encapsulated within the Access-Request Packet, acquire the data sent from the client. It should then concatenate all the data together pre-fixed sequence for items in the combination code. Then it should encrypt that data using the MAC Address as the key and AES Algorithm as the encryption algorithm to produce the ciphertext, then it should retrieve key corresponding to the MAC Address from Database and compare the ciphertext to the key retrieved. If both are same it should

then produce EAP-Success Packet and encapsulate it in an Access-Accept Packet and send it to the Authenticator. It should then return the state of System back to State 0. If the key and ciphertext are different it should then produce EAP-Failure Packet and encapsulate it in an Access-Reject Packet and send it to the Authenticator. It should then return the state of System back to State 0.

4. When the Plugin is in State 2, it receives a Continue Registration message (Can be encoded) from the Registration module, it should Produce the EAP Request/Vendor Specific Packet with MAC Address in the Vendor Specific Data part of the packet and encapsulate it in Access-Challenge Packet and send to authenticator. The Plugin should go to State 3.
5. When the Plugin receives an Access-Request Packet from authenticator, containing a MAC Address in State 3, it should parse the EAP Response/Vendor Specific Packet to obtain the information sent by the client. It should then concatenate all the data together in a prefixed order Then it should encrypt that data using the MAC Address as the key and AES Algorithm to obtain the ciphertext, then it should pass the ciphertext to the Registration module, and the plugin should go to state 4.
6. When the plugin is in state 4, and it receives Confirm Registration message (Can be encoded) from the Registration, then it should save the key generated to the Database.