AN2902

ATWINC Enterprise Security Application Note

Introduction

This application note describes the ATWINC Enterprise Security mode and demonstrates the basic Wi-Fi® connection between the device (acting as a station (STA)) and an Access Point (AP) in the Enterprise Security mode.

The references to the ATWINC include the following:

- ATWINC1500
- ATWINC1510
- ATWINC3400

Features

The ATWINC supports the following Enterprise WPA/WPA2 methods:

- EAP-PEAPv0/MSCHAPv2
- EAP-PEAPv1/MSCHAPv2
- EAP-PEAPv0/TLS
- EAP-PEAPv1/TLS
- EAP-TLS
- EAP-TTLS/MSCHAPv2

Table of Contents

1. Enterprise Security

The Enterprise mode of Wi-Fi Protected Access (WPA or WPA2) encryption uses 802.1X authentication to provide better security for wireless networks. The Enterprise mode suits all businesses and organizations rather than the Personal or Pre-Shared Key (PSK) mode. In the Enterprise mode, each client generates a unique encryption key for logging into the network, a technique which helps protect from malicious hacking.

1.1 IEEE® 802.1X

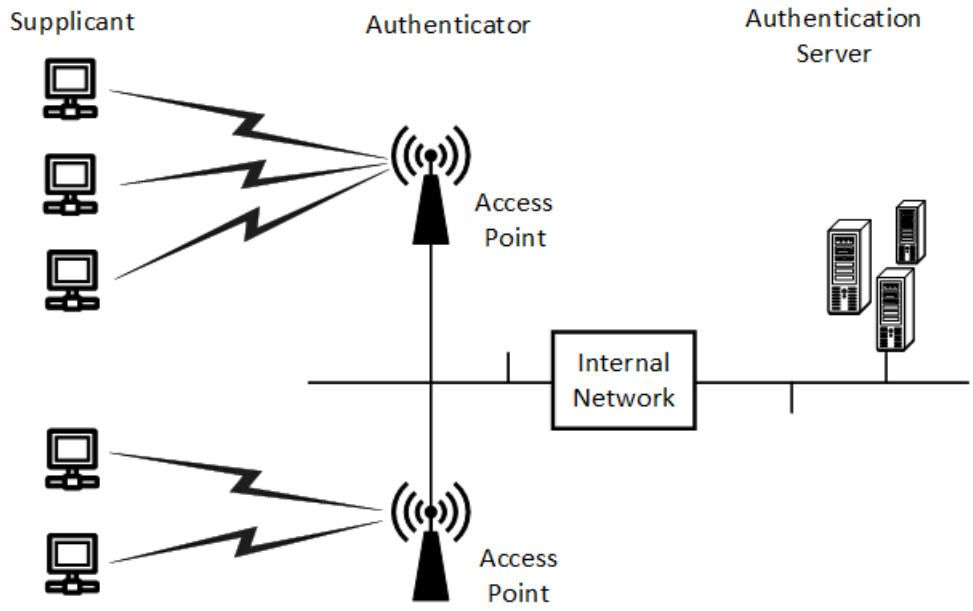
The IEEE 802.1X is a standard for port-based access control. It provides an authentication mechanism for the devices which are on a Local Area Network (LAN) or Wireless Local Area Network (WLAN).

The IEEE 802.1X authentication involves three parties: a supplicant, an authenticator and an authentication server.

- A **supplicant** is the client/end user device (station device) which tries to get authenticated by submitting the credentials such as username, password and digital certificates to an access point (authenticator). For example: a laptop, a mobile phone or the ATWINC (in the Station mode).
- An **authenticator** is a network access device which collects the authentication credentials from the supplicant, encrypts the credentials and relays those credentials to the authentication server for verification. For example: Ethernet switch or wireless access point.
- An **authentication server** is a network server which validates the credentials sent by the supplicant based on the information stored in its database and determines whether to allow or prevent network access to the supplicant. An authentication server is typically a host running software supporting the Remote Authentication Dial-In User Service (RADIUS) and Extensible Authentication Protocol (EAP) protocols.

The authentication server guards to protect the network and does not allow the supplicant for the network access unless supplicant identity is validated and authorized.

Figure 1-1. IEEE 802.1X Authentication Mechanism



The authenticator encrypts the credentials to forward to the authentication server. If an authentication server determines the credentials to be valid, the supplicant is allowed to access the network ports.

1.2 Enterprise Network

When a wireless station connects to an enterprise enabled access point, it is identified as a new supplicant. Firstly, the new supplicant connects to the access point by performing an Open System Authentication and performing the frame exchange for authentication and association. Once the Open System Authentication phase completes, the EAP authentication starts. Until the EAP authentication is completed, all other traffic to the new supplicant is blocked.

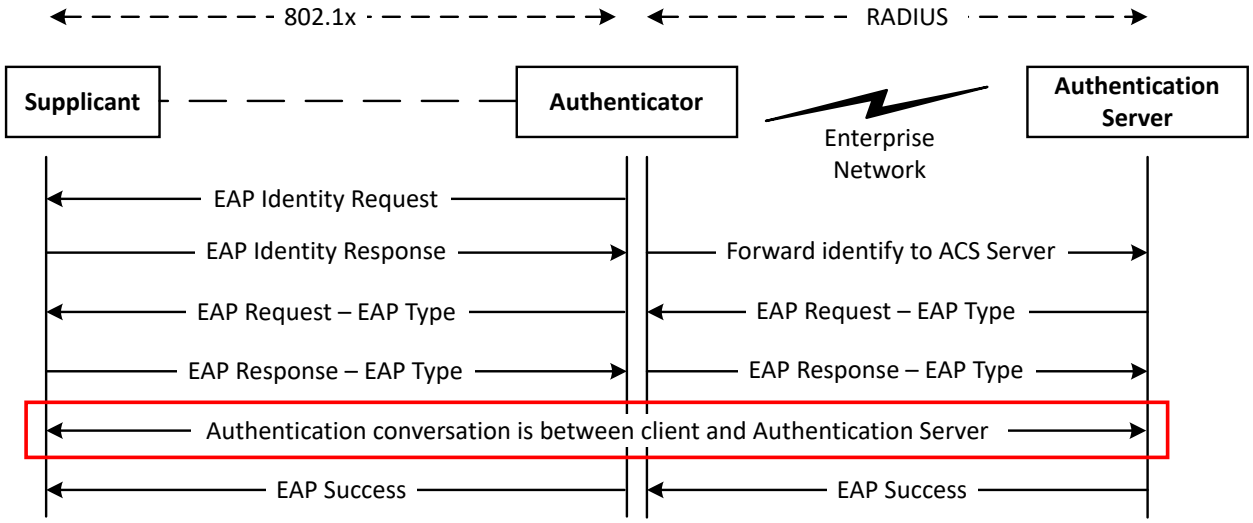
The EAP authentication starts with the authenticator sending an EAP Identity frame to the supplicant. The supplicant, on receiving the EAP request identity, responds with EAP Identity response frame containing user ID to the authenticator. Then the authenticator encapsulates this EAP identity response in a RADIUS access request packet and forwards it to the authentication server.

The authentication server sends a reply (encapsulated in a RADIUS access challenge packet) to the authenticator containing an EAP Request specifying the EAP method. The supplicant can do one of the following:

1. Use the EAP method requested by an EAP response, or,
2. Send NAK (negative acknowledgment) and respond with the EAP methods it supports.

Finally, the authentication server and the supplicant must agree on one EAP method to proceed with the authentication process. Based on the EAP method, EAP requests and EAP responses are sent between supplicant and authentication server until the authentication server responds with EAP-Success or EAP failure packet. If the authentication is successful, the authenticator allows normal traffic to the supplicant. If authentication is unsuccessful, the authenticator blocks all other traffic (except EAP data frames) to the supplicant.

Figure 1-2. Enterprise Network Flow Diagram



During EAP authentication, the supplicant and the authentication server derive a Pairwise Master Key (PMK) for data encryption. This key is unique for each session of a given client. For broadcast and multicast traffic it uses a Group Transient Key (GTK) which is common to all clients. The authentication server sends the derived PMK to the authenticator, and the supplicant and the authenticator perform a four-way handshake to complete the authentication process.

1.3 Extensible Authentication Protocol (EAP)

The Extensible Authentication Protocol (EAP) is a point-to-point (P2P) wireless and LAN authentication framework providing a variety of authentication mechanisms. The EAP method provides a request or response framework over which a specific authentication algorithm is implemented. Most commonly used EAP methods in wireless networks are EAP-TLS, EAP-PEAPv0, EAP-PEAPv1 and EAP-TTLS. The following figure shows the summary of the EAP packet format. The fields read from left to right.

Figure 1-3. EAP Packet Format

Code 1 byte	Identifier 1 byte	Length 2 bytes	Data (0 to n bytes)	
			Type	Type Data

Code – This has 8 bits. It identifies the type of the EAP packet and can have the following EAP code numbers:

- 1 – Request
- 2 – Response
- 3 – Success
- 4 – Failure

Identifier – This has 8 bits and matches Responses with Requests

Length – This field is 16 bits and indicates the length, in octets, of the EAP packet including the Code, Identifier, Length and Data fields.

Data – The format of this field is determined by the Code field.

If the code is set to Request/Response, the Data field consists of a byte which indicates the EAP Type, followed by zero or more bytes of Type Data.

The EAP Types recognized by the ATWINC Enterprise implementation are:

- 1 – Identity
- 3 – Nak
- 13 – TLS
- 21 – TTLS
- 25 – PEAP
- 26 – MSCHAPv2
- 33 – Extensions (used within PEAPv0 only)

For the official registry of all EAP Types, refer to www.iana.org/assignments/eap-numbers/eap-numbers.xhtml.

Note: For more details about EAP protocol, refer to tools.ietf.org/html/rfc3748.

1.4 EAP Methods

The EAP Authentication is a framework that provides request - response functions (for negotiation and authentication) that implement a specific authentication algorithm called EAP Method.

The ATWINC supports the following EAP Methods:

1. EAP Transport Layer Security (EAP-TLS)
2. EAP Tunneled Transport Layer Security (EAP-TTLS)
3. EAP Protected Extensible Authentication Protocol (EAP-PEAP)

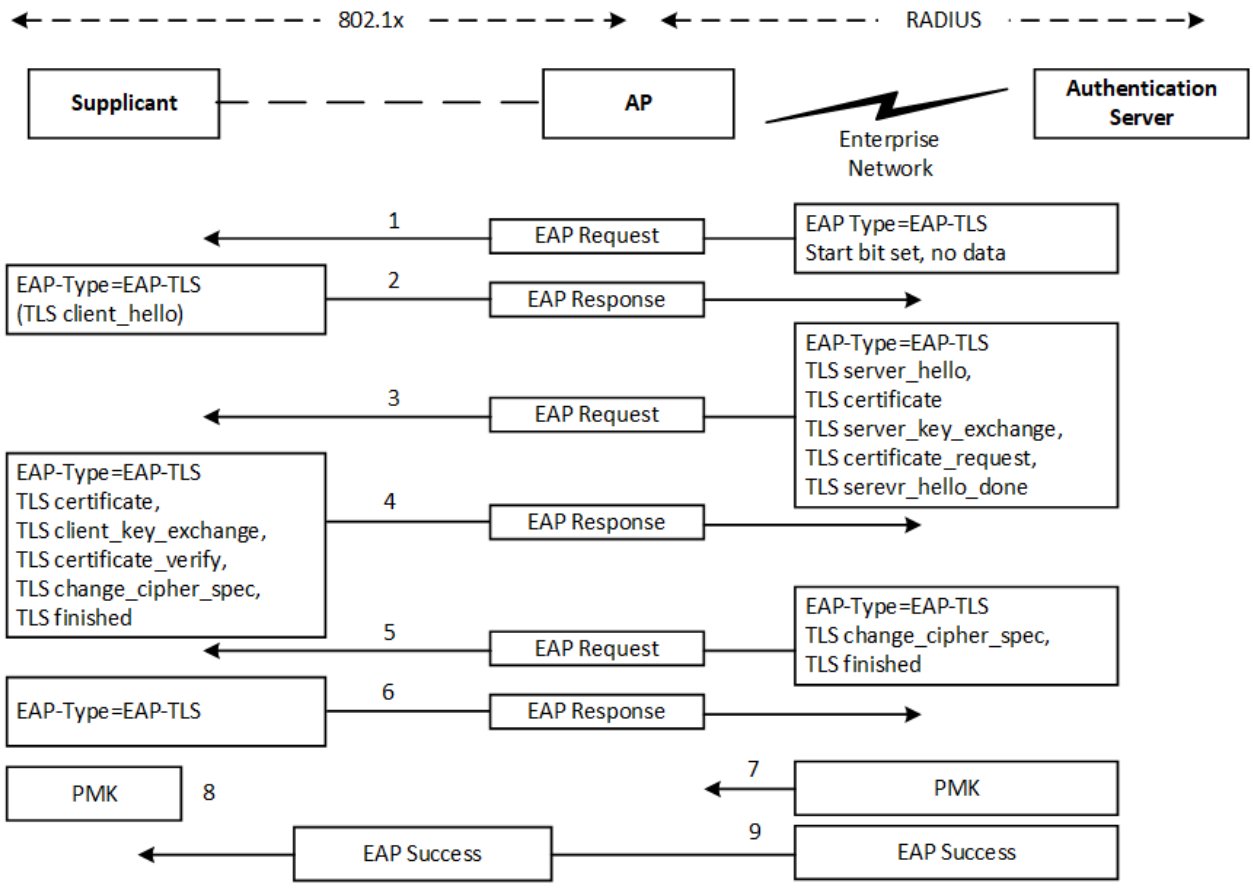
1.4.1 EAP-TLS (Transport Layer Security)

The EAP-TLS (RFC 5216) uses the TLS protocol (RFC 5246), which is the Internet Engineering Task Force's (IETF®) latest version of the Secure Socket Layer (SSL) protocol. TLS provides a way to use certificates for both user and server authentication and for dynamic session key generation.

1. The EAP-TLS conversation typically begins with the authenticator and the peer negotiating EAP. The EAP server must respond with an EAP-TLS/Start packet, which is an EAP-Request packet with EAP-Type = EAP-TLS, the Start(S) bit is set and no data.
2. The EAP-TLS conversation, then, begins with the peer sending an EAP-Response packet with EAP-Type = EAP-TLS. The data field of that packet encapsulates one or more TLS records in TLS record layer format, containing a TLS client_hello handshake message.
3. The EAP server, then, responds with a server_hello handshake message, TLS certificate, server_key_exchange, certificate_request, server_hello_done and/or finished handshake messages and/or a TLS change_cipher_spec message.

4. The Client must respond to the EAP-Request with an EAP-Response packet of EAP-Type = EAP-TLS. The data field must encapsulate one or more TLS records containing a TLS certificate, TLS certificate verify, TLS client_key_exchange, change_cipher_spec and TLS finished message.
5. If a ChangeCipherSpec message is sent by the client and the client requests to switch to symmetric key encryption, the server responds with its own ChangeCipherSpec message to confirm the switching to symmetric key encryption and sends its TLS finished message under the new Cipher Spec. For more information, refer to https://tools.ietf.org/html/rfc5246.
6. If the EAP server authenticates successfully, the peer must send an EAP-Response packet of EAP-Type = EAP-TLS and no data.
7. The authentication server and the supplicant each derive the PMK (from the material exchanged during the TLS handshake).
8. The authentication server sends the PMK to the authenticator (AP).
9. The EAP server, then, must respond with an EAP-Success message.

Figure 1-4. EAP-TLS Protocol Method



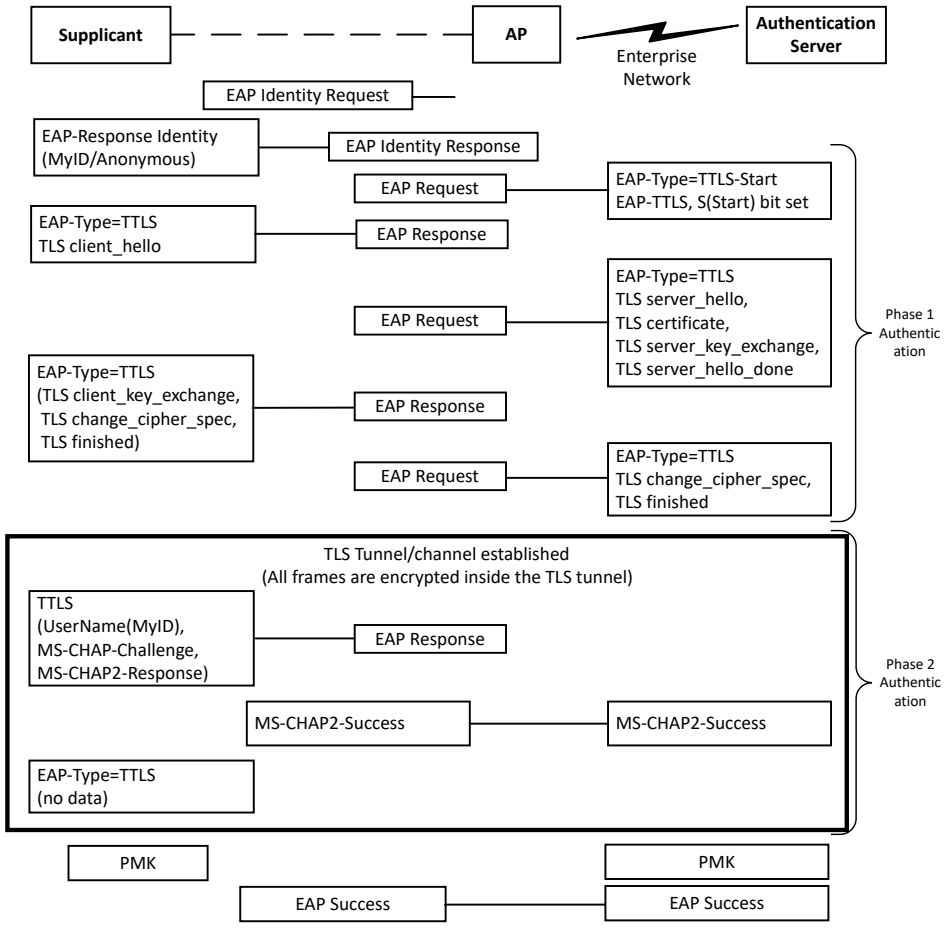
1.4.2 EAP-TTLS

In the EAP-TLS, a TLS handshake is used to mutually authenticate a client and server, whereas, with EAP-TTLS (RFC 5281), the TLS handshake authenticates the server and not the client. The client is authenticated by another method, which takes place inside the secure tunnel established by the TLS handshake. There are two phases in EAP-TTLS, the TLS handshake phase (Phase 1) and the TLS tunnel phase (Phase 2).

- In the handshake phase, the server is authenticated to the client using the standard TLS procedure, and keying material is generated to create a cryptographically secure tunnel for information exchange in the subsequent data phase.

- In the tunnel phase, the TLS record layer is used to securely tunnel information between the client and the TTLS server. In this phase, the client is authenticated to the server using an arbitrary authentication mechanism encapsulated within the secure tunnel.
- The encapsulated authentication mechanism may itself be EAP or it may be another authentication protocol such as PAP, CHAP, MS-CHAP or MSCHAP-V2 (ATWINC supports only MSCHAP-V2).

Figure 1-5. EAP-TTLS Protocol Method



1.4.3 EAP-PEAP (Protected Extensible Authentication Protocol)

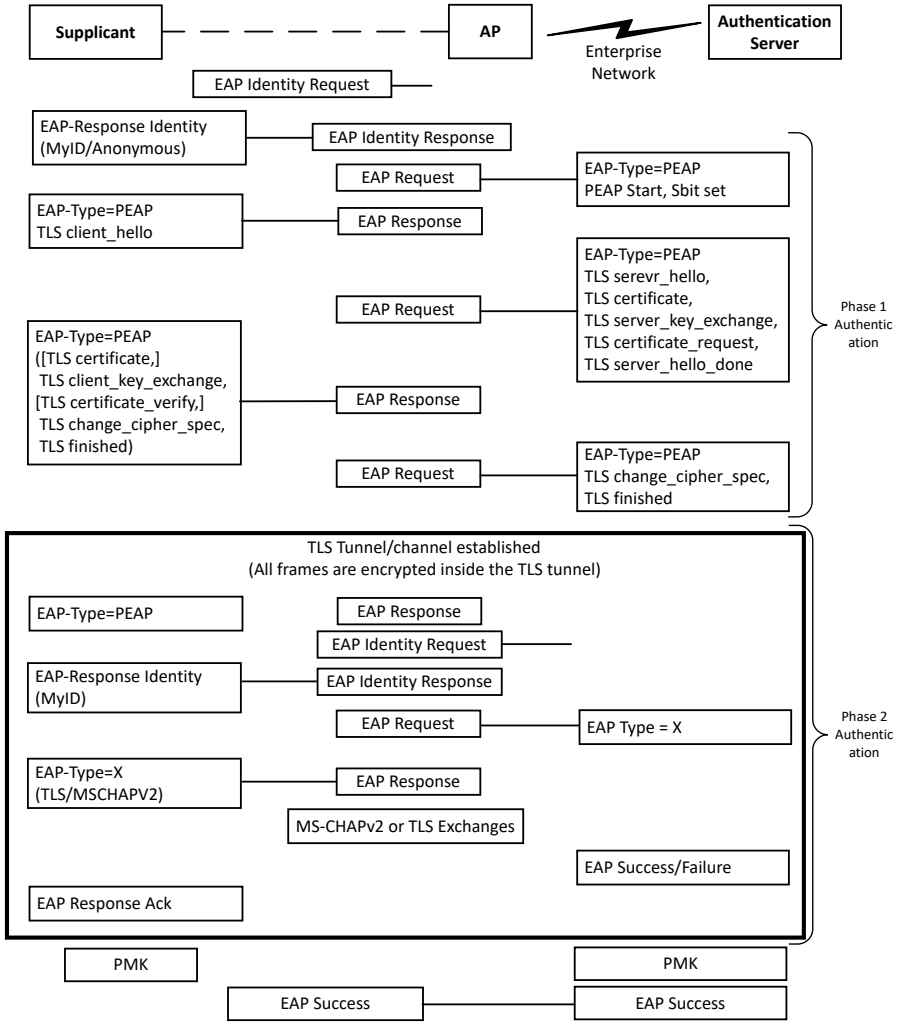
The Protected Extensible Authentication Protocol (PEAP), also known as Protected EAP or simply PEAP, is a protocol that encapsulates EAP within a potentially encrypted and authenticated Transport Layer Security (TLS) tunnel.

The PEAP operates in two phases.

- Phase 1 – EAP peer establishes a TLS session and authenticates with the EAP server.
- Phase 2 – An inner method is negotiated over the TLS session of Phase 1.

There are different versions of PEAP. The ATWINC implements PEAPv0 (RFC draft-kamath-pppext-peapv0-00) and PEAPv1 (RFC draft-josefsson-pppext-eap-tls-eap-05). For Phase2 authentication, the ATWINC supports MSCHAPv2 or TLS. The following figure shows the PEAPv1 authentication process. For PEAPv0 and PEAPv1, the phase1 authentication is similar. For phase2, the format of EAP messages inside the tunnel is different for PEAPv0 and PEAPv1.

Figure 1-6. EAP-PEAP Method



The PEAP is based on server side EAP-TLS authentication. With PEAP the issues associated with installing digital certificates can be avoided on every client device as required by EAP-TLS. The user can select the methods of client authentication, such as logon passwords or OTPs, which best suit their corporate needs. PEAP is an enhancement of EAP-TLS authentication, and encapsulates a second-phase authentication transaction within the TLS framework.

1.4.3.1 EAP-PEAP TLS

The phase 1 authentication is the same as in EAP-PEAP. The second phase of the PEAP conversation consists of another complete EAP-TLS conversation (as shown in Figure 1-6) occurring within the TLS session negotiated in the PEAP phase 1. Since all packets sent within the PEAP phase 2 conversation occur after TLS session establishment, they are protected using the negotiated TLS cipher suite.

1.4.3.2 EAP-PEAP MSCHAPv2

The phase 1 authentication is the same as EAP-PEAP. In phase 2 another EAP conversation occurs along with exchange of username and password as shown in Figure 1-6. All the packets in phase two are encrypted with secured TLS tunnel.

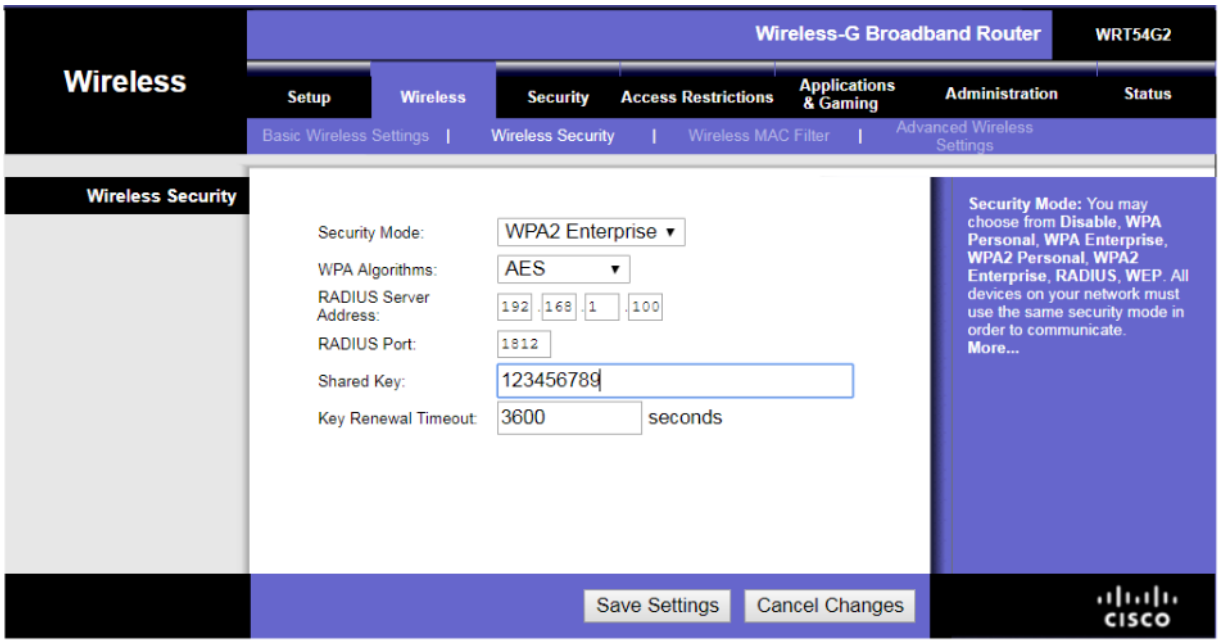
2. Authenticator - AP Configuration

The authenticator is a network device like an Ethernet switch or access point. The supplicant provides the authenticator with the username and either password or digital certificates. The authenticator forwards them to the authentication server for authorization. A typical authenticator (AP) configuration page displays in the following figure.

The following is a sample for an authenticator (AP) configuration:

- Select Security Mode as **WPA2 Enterprise**
- Enter the IP address of the RADIUS device
- Enter the RADIUS port as **1812** (Default port address for NPS)
- Enter the Shared key
- Save the settings

Figure 2-1. Authenticator - AP Configuration



3. Configuring an Authentication Server

An authentication server is a network server that validates the credentials sent by the supplicant based on the information stored in its database and determines whether to allow network access or prevent network access to the supplicant.

The most common Authentication server or Radius server used for deployment and testing are FreeRADIUS and Hostapd Server. The following sections explain how to configure a Hostapd Server and FreeRADIUS Server.

To configure a RADIUS server, the user must have the generated server certificate, client certificate and root certificate. The following section explains how to generate a root certificate using OpenSSL.

3.1 Generating Certificates using `openssl`

After installing OpenSSL, open a CMD prompt and navigate to the directory where OpenSSL is installed. Perform the following steps to generate server key, public certificate, Certificate Signing Request (CSR) and root certificate.

3.1.1 Generating Server Key

Generate a Server key using the openssl genrsa -out server.key 2048 command.

3.1.2 Generating the CA Certificate

Perform the following steps to generate the CA certificate:

1. Generate the CA key using the openssl genrsa -out winc_root.key 2048 command.
2. Generate the CA certificate using the CA key and using the openssl req -new -x509 -days 365 -key winc_root.key -out winc_root.crt command.
3. The ATWINC root certificate downloader accepts the certificates in `.der` format only. Therefore, convert the CA certificate to `.der` format using the openssl x509 -outform der -in winc_root.crt -out winc_root.cer command.

Notes:

1. To flash the root certificate onto ATWINC1500 Flash, save the `winc_root.cer` file in the root certificate downloader folder `\firmware_update_project\firmware\Tools\root_certificate_downloader\binary` in the firmware update project and perform the firmware update.
2. If the certificate upload fails with “(ERROR) Root Certificate Flash is Full,” then, the ATWINC memory for certificates is full and the user must upload the certificate after removing one or more certificates from `src\firmware\Tools\root_certificate_downloader\binary` folder.
3. For more details, refer to the *WINC1500/WINC3400 Integrated Serial Flash Memory Download Procedure* (DS00002378) document.

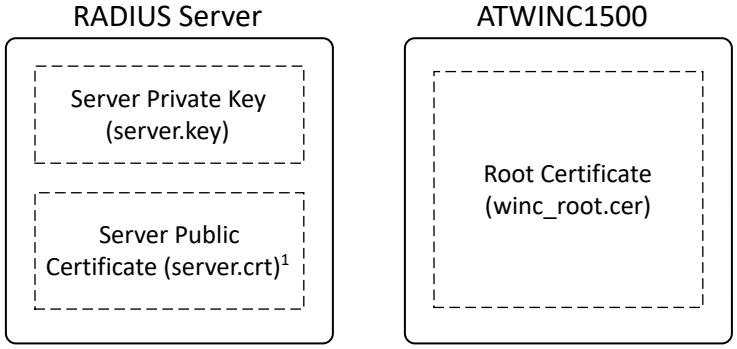
3.1.3 Generating a Certificate Signing Request and a Public Certificate

Perform the following steps to generate the Certificate Signing Request (CSR) and public certificate:

1. Generate the CSR using the server key (`server.key`) and using the openssl req -new -key server.key -out server.csr command.
2. Self-sign the certificate using the CA certificate and generate the public key using the openssl x509 -req -days 365 -in server.csr -CA winc_root.crt -CAkey winc_root.key -set_serial 01 -out server.crt command.

   The above-generated certificates (`server.crt`, `server.key` and `winc_root.cer`) are used for server authentication. During server authentication, `server.crt` and `server.key` are used by the RADIUS server. The root certificate `winc_root.cer` is flashed into the ATWINC using the root certificate downloader.

Figure 3-1. Certificates Required for EAP-TTLS with MSCHAPv2 and EAP-PEAPv0/1 MSCHAPv2



1. `server.crt` must be signed by `winc_root.cer`

Notes:

- Server authentication requires `server.key` and `winc_root.cer` certificates.
- Client authentication does not use a certificate.

3.1.4 Certification Creation for TLS Client Authentication

For EAP-TLS and PEAPv0/1 with TLS, one more set of certificates is required for client authentication. Perform the following steps to generate this extra set of certificates. The ATWINC uses the newly created public certificate and server key certificate (for example, `winc_client_private.crt` and `winc_client_private.key`), and the Authentication server uses the newly created CA certificate (for example, `radius_root.crt`).

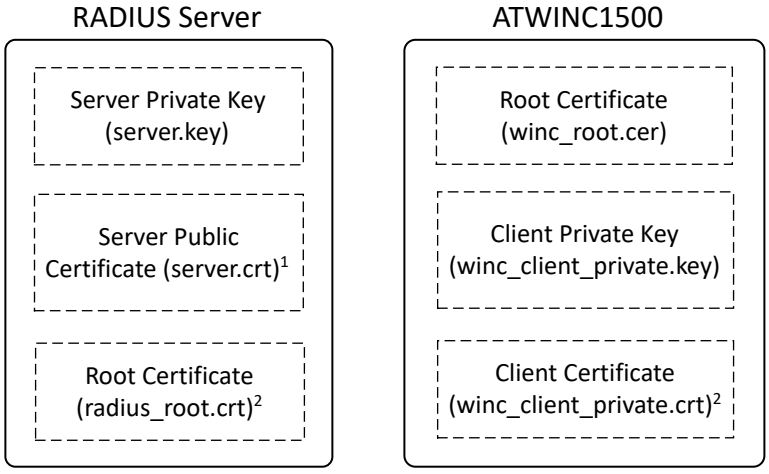
Perform the following steps to generate the certificates for TLS client authentication:

1. Generate the CA key using the openssl genrsa -out radius_root.key 2048 command.
2. Generate the CA certificate using the CA key and using the openssl req -new -x509 -days 365 -key radius_root.key -out radius_root.crt command.

Perform the following steps to generate the Certificate Signing Request (CSR) and public certificate:

1. Generate a Client key using the openssl genrsa -out winc_client_private.key 2048 command.
2. Generate the CSR using `winc_client_private key` (`client.key`) and the openssl req -new -key winc_client_private.key -out winc_client_private.csr command.
3. Self-sign the certificate using the CA certificate and generate the public key using the openssl x509 -req -days 365 -in winc_client_private.csr -CA radius_root.crt -CAkey radius_root.key -set_serial 01 -out winc_client_private.crt command.

Figure 3-2. Certificates Required for EAP-TLS and EAP-PEAPv0/1 with TLS



1. `server.crt` must be signed by `winc_root.cer`.

2. `winc_client_private.crt` must be signed by `radius_root.crt`.

Notes:

- Server authentication requires the `server.key`, `server.crt` and `winc_root.cer` certificates.
- Client authentication requires the `radius_root.crt`, `winc_client_private.key` and `winc_client_private.crt` certificates.

3.2 Configuring a Hostapd Server

Perform the following steps to configure the hostapd server:

1. Download hostapd from https://w1.fi/releases/hostapd-2.6.tar.gz and copy it to an Ubuntu machine.
2. Create a `.config` file enabling hostapd as a RADIUS server. See 7. Appendix B - Hostapd Example .config File for an example configuration file.
3. Untar the file and navigate to the `hostapd-2.6/hostapd` directory in the terminal window.
4. Build the binaries using the `make` command.
5. Enter the `make install` command to copy the `hostapd` binary to the `/user/local/bin/` path.
6. Generate the certificates (see 3.1 Generating Certificates using openssl).
7. Add the following to configure or create AP details in the file `hostapd.radius_clients` (the password must be the same as the shared key password in point 4).

```
# RADIUS client configuration for the RADIUS server
0.0.0.0/0      123456789
```

8. Create an `eap` user file (see 8. Appendix C - Configuring EAP User File).
9. Create a `hostapd.conf` file, using the above `eap` user file as shown below.

```
# Run hostapd as a RADIUS server
radius_server_clients=hostapd.radius_clients
radius_server_auth_port=1812

eap_server=1
# For EAP user file see section 5.3
eap_user_file=hostapd.eap_user

# TLS parameters (shared by EAP-PEAP, EAP-TTLS, EAP-FAST)
ca_cert=cas.cert
# Server certificate and private key from separate files
server_cert=server.crt
private_key=server.key
```

10. Run `hostapd` using the sudo ./hostapd -dkt -i eno1 hostapd.conf command.

3.3 Configuring a FreeRADIUS Server

Perform the following steps to configure the FreeRADIUS server:

1. Download and install the RADIUS server 3.x version on a Linux® machine.
2. Modify the text `allow_vulnerable_openssl = no` in `/usr/local/etc/raddb/radiusd.conf` to the following:

```
"allow_vulnerable_openssl= 'CVE-2016-6304'
```

3. Open the file `/usr/local/etc/raddb/client.conf` and provide the same AP IP address and shared key as mentioned in 2. Authenticator - AP Configuration.

```
For Example:
        client WINC1500 {
            ipaddr = 192.168.1.1
            secret = 123456789
        }
```

4. Generate the certificates and keys as mentioned in the 3.1 Generating Certificates using openssl and copy to the `/usr/local/etc/raddb/certs` path.

5. Select EAP security for phase 1 authentication in `/usr/local/etc/raddb/mods-available/eap` file and modify the following in the EAP mode.
   – For TTLS
   ```
   default_eap_type = ttls
   ```
   – For TLS
   ```
   default_eap_type = tls
   ```
   – For PEAP
   ```
   default_eap_type = peap
   ```
6. Search for the string `tls-config tls-common` in the `/usr/local/etc/raddb/mods-available/eap` file and map the proper key file and certificate file as shown below. This is common for TLS, TTLS and PEAP.
   ```
   private_key_file = ${certdir}/server.key
    certificate_file = ${certdir}/server.crt
    ca_file = ${cadir}/radius_root.crt
   ```
7. For phase 2 authentication.
   – For TTLS in ttls mode
   ```
   default_eap_type = mschapv2
   ```
   – For PEAP in peap mode
   ```
   default_eap_type = mschapv2
   ```
8. Configure the EAP users for the phase 2 authentication in the file `mods-config/files/authorize` used for MSCHAPv2.
   ```
   DEMO_USER Cleartext-Password := “DemoPassword”
   DEMO_AP Cleartest-Password := “12345678”
   ```
9. Run the RADIUS server using the `radius -x` command.

4. ATWINC Host APIs

The following table lists the APIs that are available in the application for requesting a connection to an Enterprise network.

Table 4-1. ATWINC Host APIs

API	Description
`m2m_wifi_connect_1x_tls`	Connects to an Enterprise network using TLS client credentials. The full authentication method (EAP-TLS, EAP-PEAPv0/TLS or EAP-PEAPv1/TLS) depends on the configuration of the authentication server.
`m2m_wifi_connect_1x_mschap2`	Connects to an Enterprise network using MSCHAPv2 credentials. The full authentication method (EAP-TTLSv0/MSCHAPv2, EAP-PEAPv0/MSCHAPv2 or EAP-PEAPv1/MSCHAPv2) depends on the configuration of the authentication server.
`m2m_wifi_default_connect`	Reconnects to the last connected Enterprise network (assuming a previous connection request used the option to store the credentials in the ATWINC Flash).

5. ATWINC Applications

This section provides the examples for connecting the ATWINC to a TLS secured AP, MSCHAPv2 secured AP and for the ATWINC Enterprise Security Provisioning application. The examples are available in ASF3 (v3.42 and above).

5.1 Example 1 - Connecting ATWINC to TLS Secured AP

The EAP-TLS authentication is based on the 802.1x/EAP architecture. The 802.1x/EAP authentication process involves the following components:

1. Supplicant (ATWINC)
2. Authenticator (wireless access point configured for Enterprise security)
3. Authentication server (RADIUS server or PC with FreeRADIUS or Hostapd installed)

Perform the following steps to connect the ATWINC using the EAP-TLS enterprise security:

1. In Atmel Studio, open the WINC1500_SECURITY_ENTERPRISE_NETWORK_TLS_EXAMPLE project.
2. Configure and run the FreeRADIUS or hostapd server (see 3.2 Configuring a Hostapd Server and 3.3 Configuring a FreeRADIUS Server).
3. Provide the macro MAIN_WLAN_802_1X_USR_NAME (EAP username).
4. Flash the root certificate to the ATWINC. For more details, see 3.1 Generating Certificates using openssl. Ensure that the firmware and the host driver are both version v19.6.1 or above.
5. For Client authentication, download the Client private key (`winc_client_private.key`) and Client certificate (`winc_client_private.crt`) to the ATWINC. For this, decode the certificate and key files using script `key_decoder.py` and load the files through the example code.
   - The decoder script is located at `src\script\key_decoder.py`. Rename the server certificate and key files to `demo_rsa.crt` and `demo_rsa.key` because the script assumes these file names are input.
   - Run `key_decoder.py` to generate the `privateKey_decoded.txt` file.
   - Replace the modulus, exponent and certificate arrays of `main.h` with the respective `privateKey_decoded.txt` arrays. Verify the length of the arrays.
6. Configure the SSID by editing the macro `MAIN_WLAN_SSID` in the project.
7. Configure and run the FreeRADIUS or hostapd server (see 3.2 Configuring a Hostapd Server and 3.3 Configuring a FreeRADIUS Server).
8. Load the example project.

Notes: The `key_decoder.py` Python® script requires the pycrypto package, which depends on Visual C++® 9. Therefore, install the Visual C++ 9 using the following steps:

1. Go to the link aka.ms/vcpython27 and install the pycrypto package.
2. Enter the pip install pycrypto command.

5.2 Example 2 - Connecting ATWINC to MSCHAPv2 Secured AP

Perform the following steps to connect the ATWINC using the MSCHAPv2 enterprise security:

1. In the Atmel Studio, open the WINC1500_SECURITY_ENTERPRISE_NETWORK_MSCHAPV2_ EXAMPLE project.
2. Configure and run the FreeRADIUS or hostapd server (see 3.2 Configuring a Hostapd Server and 3.3 Configuring a FreeRADIUS Server).
3. For server authentication, the root certificate must be downloaded to the ATWINC. For more details, see 3.1 Generating Certificates using openssl.
4. Flash the root certificate to the ATWINC.
5. Provide the macros `MAIN_WLAN_802_1X_USR_NAME` (EAP username) and `MAIN_WLAN_802_1X_PWD` (EAP password).
   - For the hostapd server, see 8. Appendix C - Configuring EAP User File for the EAP username and password.

– For the FreeRADIUS server, the username and password are available in the file `mods-config/files/authorize`.

6. Configure the SSID by editing the macro `MAIN_WLAN_SSID` in the project.
7. Load the example project.

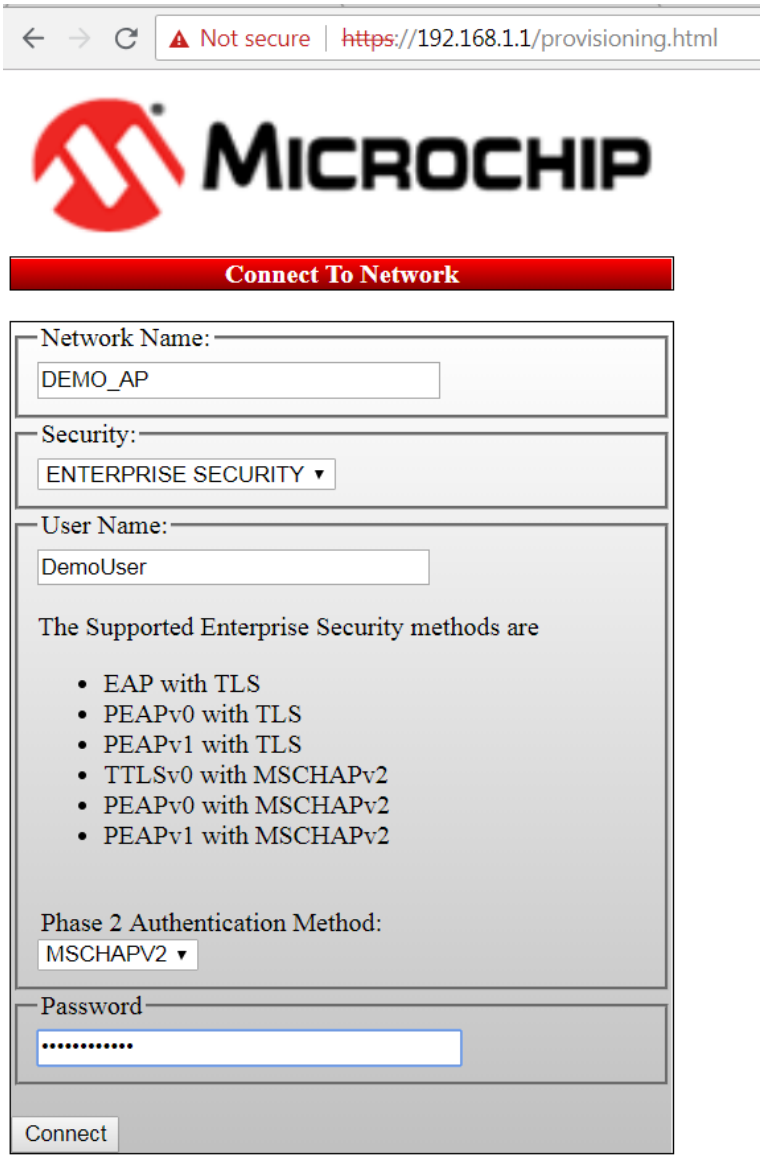
5.3 Example 3 - Launching ATWINC Enterprise Security Provisioning Application

In the provisioning example, initially the ATWINC enumerates as a soft AP with the SSID provided by the parameter `PROV_WLAN_SOFTAP_SSID` in the file `wifi_prov.h`.

Perform the following steps to launch the ATWINC Enterprise Security Provisioning application:

1. Connect the laptop/mobile to the enumerated soft AP.
2. Once the Wi-Fi link is established, open the Google Chrome™ or Firefox® web browser and open the following web page: 192.168.1.1/provisioning.html.
3. Enter the credentials of the AP that the ATWINC must be connected to.

Figure 5-1. ATWINC Enterprise Security Provisioning Application



4. Click **Connect**.

5.3.1 Changing the Logo of ATWINC Enterprise Security Provisioning Application

Perform the following steps to change the logo of the ATWINC Enterprise Security Provisioning application:

1. Open the project WINC1500_SECURITY_ENTERPRISE_PROVISIONING_EXAMPLE from ASF.
2. Navigate to `src\ASF\common\components\wifi\winc1500\host_app\provisioning\script` in the example project and replace the available logo with the required logo.
3. Convert the logo and html content (web page) to HEX format by running the `hexdump.py` script. It generates the `html_logo_c_array.txt` file which has `html_buff` and `logo_buff` arrays.
4. Copy the content of the arrays `html_buf` and `logo_buff` to the file `html_page_buf.h` located at `src\ASF\common\components\wifi\winc1500\host_app\provisioning`.
5. Build and load the example.

6. Appendix A - Debugging Logs

This section provides the debug UART log for EAP-TLS and EAP-TTLS/MSCHAPv2.

6.1 Debug UART Log for EAP-PEAPv0/TLS

```
(0)MAC:efuse
(0)MAC_ADDR  = F8:F0:05:F4:32:34
(0)Shr_buf static: 0, 5, 5, 22, 9, 10
(10)NMI M2M SW  VER 19.6.1 REV 16761
(10)NMI MIN DRV VER 19.3.0
(10)FW URL branches/rel_1500_19.6.1
(10)Built May 23 2018   14:39:16
(10)ROM VER_2
(10)__HW_AES__
(20)(M2M)LOAD SEC
(20)(TLS)TLS Sess Sz=1572
(40)PSM off
(60)(M2M)LOAD CON
(70)(M2M)Wifi Connect
(70)(M2M)SSID: ENT_TEST
(70)(M2M)BSSID: 00:00:00:00:00:00
(70)(M2M)AUTH: WPA-Enterprise
(70)(M2M)FastCh: 1
(80)(M2M)LOAD SEC
(80)Reset MAC
(90)(GP_REG)USE PMU
(90)AIC CORR (FW) = 17d1
(90)PIC CORR (FW) = fb9
(100)PSM on
(100)MAC State <3>
(100)Set Fast Ch 1
(160)MAC State <4>
(160)MAC State <3>
(300)MAC State <4>
(300)Fast conn, Rssi -31 Ch 13
(310)Join on 13 ENT_TEST Bss 94:10:3e:c6:d6:c1 Rssi -31
(310)MAC State <5>
(310)MAC State <6>
(310)MAC State <7>
(310)MAC State <9>
(310)MAC State <10>
(310)(EAP)Stop
(310)MAC State <1>
(310)(EAP)<- Start
(320)(M2M)LOAD TLS
(320)(EAP)-> Layer:0 Code:1 Type:1
(320)(EAP)<- Layer:0 Type:1
(330)(EAP)-> Layer:0 Code:1 Type:25
(330)(TLS)Creating EAP
(330)()<- ClientHello
(330)(EAP)<- Layer:0 Type:25
(340)(EAP)-> Layer:0 Code:1 Type:25
(340)()-> ServerHello
(340)>> TLS_RSA_WITH_AES_128_GCM_SHA256
(340)()-> Certificate
(350)(TLS)*=*=* X509 *=*=*
(350)(TLS)      Subject < >
(350)(TLS)      Issuer  < >
(350)(TLS)      <2017-04-20 12:11:52> to <2027-04-18 12:11:52>
(350)(TLS)Root Cert <RSA>
(350)(TLS)      <2017-04-20 12:11:52> to <2027-04-18 12:11:52>
(360)(TLS)Root Valid
(360)()-> ServerHelloDone
(410)Tsf join
(430)()<- ClientKeyExchange
(440)()<- ChangeCipherSpec
(440)()<- Finished
(440)(EAP)<- Layer:0 Type:25
(450)(EAP)-> Layer:0 Code:1 Type:25
(460)()-> ChangeCipherSpec
```

```
(460)()-> ServerFinished
(460)(TLS)Sess() Established==>TLSv1.2
(470)(EAP)<- Layer:0 Type:25
(470)(EAP)-> Layer:0 Code:1 Type:25
(470)(EAP)-> Layer:1 Code:1 Type:1
(480)(EAP)<- Layer:1 Type:1
(480)(EAP)<- Layer:0 Type:25
(480)(EAP)-> Layer:0 Code:1 Type:25
(490)(EAP)-> Layer:1 Code:13 Type:13
(490)(TLS)Creating EAP
(490)()<- ClientHello
(490)(EAP)<- Layer:1 Type:13
(500)(EAP)<- Layer:0 Type:25
(510)(EAP)-> Layer:0 Code:1 Type:25
(530)Tsf join Done
(530)(EAP)-> Layer:1 Code:13 Type:13
(530)()-> ServerHello
(530)>> TLS_RSA_WITH_AES_128_GCM_SHA256
(530)()-> Certificate
(540)(TLS)*=*=* X509 *=*=*
(540)(TLS)      Subject < >
(540)(TLS)      Issuer  < >
(540)(TLS)      <2017-04-20 12:11:52> to <2027-04-18 12:11:52>
(540)(TLS)Root Cert <RSA>
(540)(TLS)      <2017-04-20 12:11:52> to <2027-04-18 12:11:52>
(540)(TLS)Root Valid
(540)(EAP)<- Layer:1 Type:13
(550)(EAP)<- Layer:0 Type:25
(550)(EAP)-> Layer:0 Code:1 Type:25
(560)(EAP)-> Layer:1 Code:13 Type:13
(560)()-> CertificateRequest
(560)()-> ServerHelloDone
(640)()<- Certificate
(650)()<- ClientKeyExchange
(1660)()<- CertificateVerify
(1670)()<- ChangeCipherSpec
(1670)()<- Finished
(1670)(EAP)<- Layer:1 Type:13
(1690)(EAP)<- Layer:0 Type:25
(1700)(EAP)-> Layer:0 Code:1 Type:25
(1700)(EAP)<- Layer:0 Type:25
(1710)(EAP)-> Layer:0 Code:1 Type:25
(1710)(EAP)-> Layer:1 Code:13 Type:13
(1710)(EAP)<- Layer:1 Type:13
(1720)(EAP)<- Layer:0 Type:25
(1730)(EAP)-> Layer:0 Code:1 Type:25
(1730)(EAP)-> Layer:1 Code:13 Type:13
(1730)()-> ChangeCipherSpec
(1740)()-> ServerFinished
(1740)(TLS)Sess() Established==>TLSv1.2
(1740)(EAP)<- Layer:1 Type:13
(1750)(EAP)<- Layer:0 Type:25
(1750)(EAP)-> Layer:0 Code:1 Type:25
(1760)(EAP)-> Layer:1 Code:1 Type:1
(1760)(EAP)<- Layer:1 Type:33
(1760)(EAP)<- Layer:0 Type:25
(1760)(EAP)Success Ind
(1770)(EAP)Stop
(1770)(M2M)LOAD SEC
(1790)(M2M)LOAD CON
(1810)(M2M)WIFI Connected
(1810)(DHCP)<-REQ
(2310)(DHCP)<-REQ
(3310)(DHCP)<-REQ
(3830)(DHCP)->ACK
(3830)(DHCP)Self IP     : "10.100.1.101"
(11100)S HT 0 HR 5 R 22 T 0 AT 0 F 24 B 0 NPA 22 D 300 ST 0
(11330)(DHCP)DHCP REN
(11330)(DHCP)<-REQ
(11340)(DHCP)->ACK
(11340)(DHCP)Self IP     : "10.100.1.101"
```

6.2 Debug UART Log for EAP-TTLS/MSCHAPv2

```
(0)(M2M)DriverInfo: 0x13301361: 19.6.1
(0)(M2M)ChMapV(1)
(0)Chip ID = 1503a0
(0)Flash ID = 1440ef, Size = 8 MBit
(0)MAC:efuse
(0)MAC_ADDR  = F8:F0:05:F4:32:34
(0)Shr_buf static: 0, 5, 5, 22, 9, 10
(10)NMI M2M SW  VER 19.6.1 REV 16761
(10)NMI MIN DRV VER 19.3.0
(10)FW URL branches/rel_1500_19.6.1
(10)Built May 23 2018    14:39:16
(10)ROM VER_2
(10)__HW_AES__
(20)(M2M)LOAD SEC
(20)(TLS)TLS Sess Sz=1572
(40)PSM off
(60)(M2M)LOAD CON
(70)(M2M)Wifi Connect
(70)(M2M)SSID: ENT_TEST
(70)(M2M)BSSID: 00:00:00:00:00:00
(70)(M2M)AUTH: WPA-Enterprise
(70)(M2M)FastCh: 1
(80)(M2M)LOAD SEC
(80)Reset MAC
(90)(GP_REG)USE PMU
(90)AIC CORR (FW) = 17d1
(90)PIC CORR (FW) = fb9
(90)PSM on
(90)MAC State <3>
(90)Set Fast Ch 1
(120)MAC State <4>
(120)Fast conn, Rssi -18 Ch 1
(120)Join on 1 ENT_TEST Bss 94:10:3e:c6:d6:c1 Rssi -18
(120)MAC State <5>
(120)MAC State <6>
(120)MAC State <7>
(130)Tsf join
(140)MAC State <9>
(140)MAC State <10>
(150)(EAP)Stop
(150)MAC State <1>
(150)(EAP)<- Start
(150)(M2M)LOAD TLS
(150)(EAP)-> Layer:0 Code:1 Type:1
(160)(EAP)<- Layer:0 Type:1
(160)(EAP)-> Layer:0 Code:1 Type:21
(160)(TLS)Creating EAP
(160)(EAP)<- ClientHello
(160)(EAP)<- Layer:0 Type:21
(180)(EAP)-> Layer:0 Code:1 Type:21
(180)(EAP)-> ServerHello
(180)>> TLS_RSA_WITH_AES_128_GCM_SHA256
(180)(EAP)-> Certificate
(190)(TLS)*=*=* X509 *=*=*
(190)(TLS)    Subject < >
(190)(TLS)    Issuer  < >
(190)(TLS)    <2017-04-20 12:11:52> to <2027-04-18 12:11:52>
(190)(TLS)Root Cert <RSA>
(190)(TLS)    <2017-04-20 12:11:52> to <2027-04-18 12:11:52>
(190)(TLS)Root Valid
(190)(EAP)-> ServerHelloDone
(240)Tsf join Done
(270)(EAP)<- ClientKeyExchange
(270)(EAP)<- ChangeCipherSpec
(270)(EAP)<- Finished
(280)(EAP)<- Layer:0 Type:21
(300)(EAP)-> Layer:0 Code:1 Type:21
(300)(EAP)-> ChangeCipherSpec
(300)(EAP)-> ServerFinished
(310)(TLS)Sess(EAP) Established==>TLSv1.2
(330)(EAP)<- Layer:0 Type:21
(330)(EAP)-> Layer:0 Code:1 Type:21
```

```
(340)(EAP)<- Layer:0 Type:21
(340)(EAP)Success Ind
(340)(EAP)Stop
(340)(M2M)LOAD SEC
(370)(M2M)LOAD CON
(370)(M2M)WIFI Connected
(380)(DHCP)<-REQ
(390)(DHCP)->ACK
(390)(DHCP)Self IP     : "10.100.1.102"
(7890)(DHCP)DHCP REN
(7890)(DHCP)<-REQ
(7900)(DHCP)->ACK
(7900)(DHCP)Self IP     : "10.100.1.102"
```

7. Appendix B - Hostapd Example `.config` File

The following is a sample code to create `.config` file.

```
CONFIG_DRIVER_WIRED=y
CONFIG_DRIVER_NONE=y
CONFIG_EAP=y
CONFIG_EAP_MD5=y
CONFIG_EAP_TLS=y
CONFIG_EAP_MSCHAPV2=y
CONFIG_EAP_PEAP=y
CONFIG_EAP_GTC=y
CONFIG_EAP_TTLS=y
CONFIG_EAP_SIM=y
CONFIG_EAP_AKA=y
CONFIG_EAP_AKA_PRIME=y
CONFIG_EAP_PAX=y
CONFIG_EAP_PSK=y
CONFIG_EAP_PWD=y
CONFIG_EAP_SAKE=y
CONFIG_EAP_GPSK=y
CONFIG_EAP_GPSK_SHA256=y
CONFIG_EAP_FAST=y
CONFIG_WPS=y
CONFIG_WPS_UPNP=y
CONFIG_WPS_NFC=y
CONFIG_EAP_IKEV2=y
CONFIG_EAP_TNC=y
CONFIG_EAP_EKE=y
CONFIG_PKCS12=y
CONFIG_RADIUS_SERVER=y
CONFIG_IPV6=y
CONFIG_DRIVER_RADIUS_ACL=y
```

8. Appendix C - Configuring EAP User File

The following are the methods to configure the EAP user file in the RADIUS server.

TTLS MSCHAPV2 - EAP User Configuration

Enter the username and password in the `hostapd.eap_user_ttls_mschapv2` file to configure or create EAP user file as TTLS MSCHAPV2 using the following command.

```
# Phase 2 (tunneled within EAP-PEAP/TTLS/FAST) users
*TTLS
"john"    TTLS-MSCHAPV2    "123456"    [2]
"wifi-user@ttls"    TTLS-MSCHAPV2    "test%11"    [2]
```

where, `john` is the username and `123456` is the password.

TLS - EAP User Configuration

Enter the username in the `hostapd.eap_tls_user` file to configure or create EAP user file as TLS using the following command.

```
# Phase 1 users
"john"    TLS
"DEMO_USER" TLS
```

where, `john` is the username.

PEAPV0/TLS - EAP User Configuration

Enter the username in the `hostapd.eap_user` file to configure or create EAP user file as PEAPV0/TLS using the following command.

```
* PEAP [ver=0]
"john"    TLS [2]
"DEMO_USER" TLS [2]
```

where, `john` is the username.

PEAPV1/TLS - EAP User Configuration

Enter the username in the `hostapd.eap_tls_peapv1_user` file to configure or create EAP user file as PEAPV1/TLS using the following command.

```
* PEAP [ver=0]
"john"    TLS [2]
"DEMO_USER" TLS [2]
```

where, `john` is the username.

PEAPV0/MSCHAPV2 - EAP User Configuration

Enter the username and password in the `hostapd.eap_peapv0_mschapv2_user` file to configure or create EAP users file as PEAPV0/MSCHAPV2 using the following command.

```
* PEAP [ver=0]
"john"    MSCHAPV2 “123456” [2]
"DEMO_USER" MSCHAPV2 “DemoPassword” [2]
```

where, `john` is the username and `123456` is the password.

PEAPV1/MSCHAPV2 - EAP User Configuration

Enter the username and password in the `hostapd.eap_peapv1_mschapv2_user` file to configure or create EAP user file as PEAPV1/MSCHAPV2 using the following command.

```
* PEAP [ver=0]
"john"    MSCHAPV2 “123456” [2]
"DEMO_USER" MSCHAPV2 “DemoPassword” [2]
```

where, `john` is the username and `123456` is the password.

9. Document Revision History

Revision	Date	Section	Description
C	01/2021	3.1.4 Certification Creation for TLS Client Authentication	Added new section
		Example 4 - BLE Provisioning for Connecting ATWINC3400 with MSCHAPv2 Secured AP	Deleted this section
		3.1.3 Generating a Certificate Signing Request and a Public Certificate	Minor update
B	06/2019	Example 4 - BLE Provisioning for Connecting ATWINC3400 with MSCHAPv2 Secured AP	Added new section
A	01/2019	Document	Initial revision

The Microchip Website

Microchip provides online support via our website at www.microchip.com/. This website is used to make files and information easily available to customers. Some of the content available includes:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip design partner program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Product Change Notification Service

Microchip's product change notification service helps keep customers current on Microchip products. Subscribers will receive email notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, go to www.microchip.com/pcn and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Embedded Solutions Engineer (ESE)
- Technical Support

Customers should contact their distributor, representative or ESE for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in this document.

Technical support is available through the website at: www.microchip.com/support

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods being used in attempts to breach the code protection features of the Microchip devices. We believe that these methods require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Attempts to breach these code protection features, most likely, cannot be accomplished without violating Microchip's intellectual property rights.
- Microchip is willing to work with any customer who is concerned about the integrity of its code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is "unbreakable." Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Legal Notice

Trademarks

ISBN: 978-1-5224-6957-5

Quality Management System

For information regarding Microchip's Quality Management Systems, please visit www.microchip.com/quality.

Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
Corporate Office 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 Technical Support: www.microchip.com/support Web Address: www.microchip.com **Atlanta** Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455 **Austin, TX** Tel: 512-257-3370 **Boston** Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088 **Chicago** Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075 **Dallas** Addison, TX Tel: 972-818-7423 Fax: 972-818-2924 **Detroit** Novi, MI Tel: 248-848-4000 **Houston, TX** Tel: 281-894-5983 **Indianapolis** Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380 **Los Angeles** Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800 **Raleigh, NC** Tel: 919-844-7510 **New York, NY** Tel: 631-435-6000 **San Jose, CA** Tel: 408-735-9110 Tel: 408-436-4270 **Canada - Toronto** Tel: 905-695-1980 Fax: 905-695-2078	**Australia - Sydney** Tel: 61-2-9868-6733 **China - Beijing** Tel: 86-10-8569-7000 **China - Chengdu** Tel: 86-28-8665-5511 **China - Chongqing** Tel: 86-23-8980-9588 **China - Dongguan** Tel: 86-769-8702-9880 **China - Guangzhou** Tel: 86-20-8755-8029 **China - Hangzhou** Tel: 86-571-8792-8115 **China - Hong Kong SAR** Tel: 852-2943-5100 **China - Nanjing** Tel: 86-25-8473-2460 **China - Qingdao** Tel: 86-532-8502-7355 **China - Shanghai** Tel: 86-21-3326-8000 **China - Shenyang** Tel: 86-24-2334-2829 **China - Shenzhen** Tel: 86-755-8864-2200 **China - Suzhou** Tel: 86-186-6233-1526 **China - Wuhan** Tel: 86-27-5980-5300 **China - Xian** Tel: 86-29-8833-7252 **China - Xiamen** Tel: 86-592-2388138 **China - Zhuhai** Tel: 86-756-3210040	**India - Bangalore** Tel: 91-80-3090-4444 **India - New Delhi** Tel: 91-11-4160-8631 **India - Pune** Tel: 91-20-4121-0141 **Japan - Osaka** Tel: 81-6-6152-7160 **Japan - Tokyo** Tel: 81-3-6880- 3770 **Korea - Daegu** Tel: 82-53-744-4301 **Korea - Seoul** Tel: 82-2-554-7200 **Malaysia - Kuala Lumpur** Tel: 60-3-7651-7906 **Malaysia - Penang** Tel: 60-4-227-8870 **Philippines - Manila** Tel: 63-2-634-9065 **Singapore** Tel: 65-6334-8870 **Taiwan - Hsin Chu** Tel: 886-3-577-8366 **Taiwan - Kaohsiung** Tel: 886-7-213-7830 **Taiwan - Taipei** Tel: 886-2-2508-8600 **Thailand - Bangkok** Tel: 66-2-694-1351 **Vietnam - Ho Chi Minh** Tel: 84-28-5448-2100	**Austria - Wels** Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 **Denmark - Copenhagen** Tel: 45-4485-5910 Fax: 45-4485-2829 **Finland - Espoo** Tel: 358-9-4520-820 **France - Paris** Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 **Germany - Garching** Tel: 49-8931-9700 **Germany - Haan** Tel: 49-2129-3766400 **Germany - Heilbronn** Tel: 49-7131-72400 **Germany - Karlsruhe** Tel: 49-721-625370 **Germany - Munich** Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 **Germany - Rosenheim** Tel: 49-8031-354-560 **Israel - Ra'anana** Tel: 972-9-744-7705 **Italy - Milan** Tel: 39-0331-742611 Fax: 39-0331-466781 **Italy - Padova** Tel: 39-049-7625286 **Netherlands - Drunen** Tel: 31-416-690399 Fax: 31-416-690340 **Norway - Trondheim** Tel: 47-72884388 **Poland - Warsaw** Tel: 48-22-3325737 **Romania - Bucharest** Tel: 40-21-407-87-50 **Spain - Madrid** Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 **Sweden - Gothenberg** Tel: 46-31-704-60-40 **Sweden - Stockholm** Tel: 46-8-5090-4654 **UK - Wokingham** Tel: 44-118-921-5800 Fax: 44-118-921-5820