

u-connectXpress

Wi-Fi security

Application note



Abstract

Including several configuration examples, this document describes the standards, protocols, encryption techniques, frameworks and methods associated with the wireless security features supported in u-blox u-connectXpress software for ODIN-W2 and NINA-W13/W15.

Document information

Title	u-connectXpress	
Subtitle	Wi-Fi security	
Document type	Application note	
Document number	UBX-20012830	
Revision and date	R04	21-Jul-2021
Disclosure restriction	C1 - Public	

This document applies to the following products:

Product name	
ODIN-W2	Version 8.0.0 or higher
NINA-W13/15	Version 4.0.0 or higher

u-blox or third parties may hold intellectual property rights in the products, names, logos and designs included in this document. Copying, reproduction, modification or disclosure to third parties of this document or any part thereof is only permitted with the express written permission of u-blox.

The information contained herein is provided "as is" and u-blox assumes no liability for its use. No warranty, either express or implied, is given, including but not limited to, with respect to the accuracy, correctness, reliability and fitness for a particular purpose of the information. This document may be revised by u-blox at any time without notice. For the most recent documents, visit www.u-blox.com.

Copyright © u-blox AG.


Contents

Document information	2
Contents	3
1 Security overview.....	5
1.1 Common standards.....	5
1.2 Wireless security in u-blox short range modules	7
2 Standards and topologies	8
2.1 WPA2 Personal.....	8
2.2 WPA2 Enterprise (802.1X)	9
2.3 WPA3 Personal.....	11
2.4 WPA3 Enterprise.....	11
3 EAP authentication methods	12
3.1 EAP-TLS.....	12
3.1.1 EAP-TLS usage	12
3.1.2 EAP-TLS validation	12
3.2 PEAP	12
3.3 LEAP	12
4 TLS end-to-end security	14
4.1 TLS over TCP	14
4.2 TLS over TCP usage	14
4.3 TLS features supported by u-blox.....	14
4.3.1 Supported TLS versions and features	14
4.3.2 TLS cipher suites supported by ODIN-W2	14
4.3.3 TLS cipher suites supported by NINA-W13/15.....	15
4.3.4 TLS groups supported by ODIN-W2 and NINA-W13/15	16
5 Certificates and private keys	17
5.1 Certificates and encodings	17
5.1.1 X.509 file extensions	17
5.2 Operations with certificates and keys.....	18
5.2.1 How to create certificates and keys	18
5.2.2 How to view certificates	18
5.2.3 How to convert certificate types	20
6 Configuration examples	21
6.1 Personal security.....	21
6.1.1 WPA2.....	21
6.2 Enterprise security 802.11X.....	21
6.2.1 Example RADIUS server setup.....	21
6.2.2 How to upload certificates and keys to short range module	23
6.2.3 PEAP	24
6.2.4 LEAP	25
6.2.5 EAP-TLS	26

6.3 End-to-end security	26
7 Security recommendations.....	29
7.1 Personal security.....	29
7.2 Enterprise security 802.11X.....	29
7.3 TLS end-to-end security.....	29
7.3.1 TLS for cloud applications	29
7.3.2 Server authentication	29
7.3.3 Mutual authentication.....	29
7.4 Summary	29
8 Troubleshooting guide.....	31
8.1 Troubleshooting 802.1X connection issues	31
8.1.1 Troubleshooting procedure	31
8.1.2 List of typical issues.....	32
8.1.3 Logging available on RADIUS authentication server	32
8.1.4 Wireshark capture analysis.....	33
8.1.5 Contacting u-blox support about connection issues	36
Appendix	37
A PEAP connection process	37
B TLS over TCP connection	38
C Glossary	39
Related documents	40
Revision history	41
Contact.....	42

1 Security overview

u-blox short range stand-alone modules support a combination of several wireless security features that help to prevent device applications gain illicit access to your network. Wi-Fi security protects data between the station and Access Point (AP).

 The Wi-Fi security features, including the authentication, certificate, and private key management of u-blox short range modules are configured using AT commands. See also u-connectXpress AT commands manual [3].

1.1 Common standards

Figure 1 shows a high-level characterization of the most commonly used security standards.

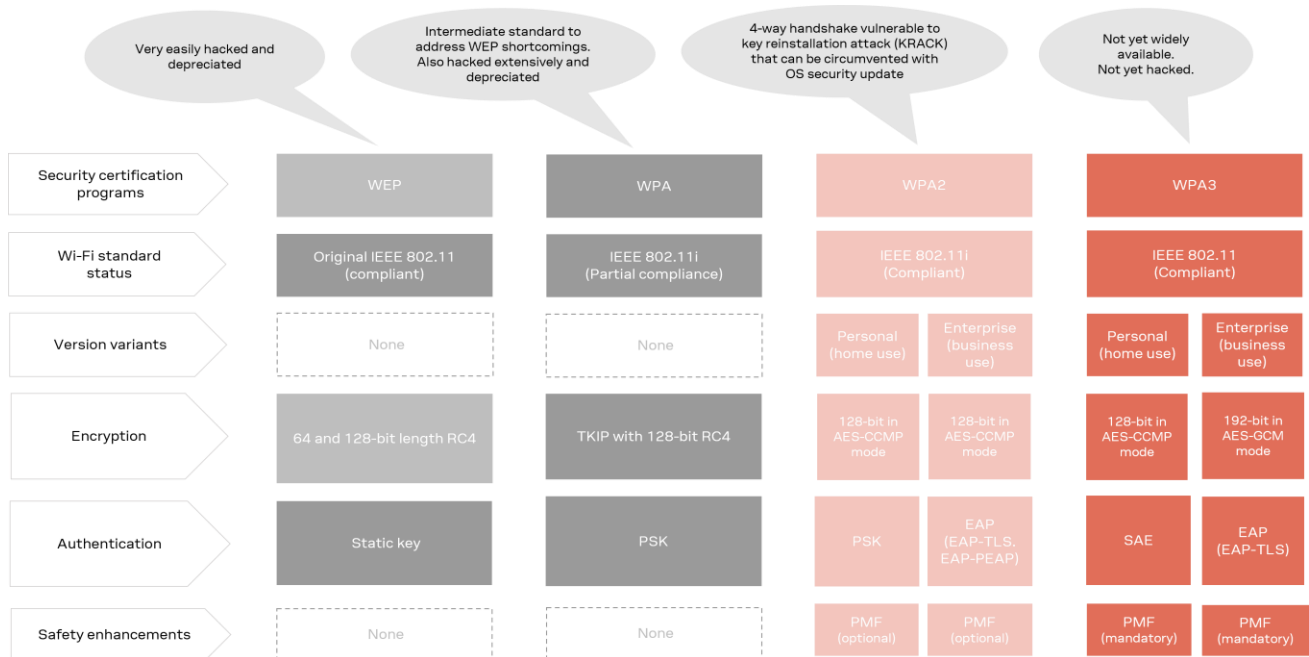


Figure 1: Commonly used security protocols – overview

WEP

- Wired Equivalent Privacy
- Superseded by WPA
- All device traffic is authenticated with a single, static encryption key (although some manufacturers leverage 802.1X authentication to achieve dynamic WEP encryption)
- WEP 64 and 128-bit key length – comprising a 40-bit shared-secret key concatenated with a 24-bit initialization vector (IV) within the 64-bit key. Many equipment suppliers extend the shared secret key to 104 bits that is combined with IV to create the 128-bit key.
- Extensive changes required to upgrade APs to WPA
- Uses RC4 (Rivest Cipher 4) encryption algorithm – now prohibited for use in TLS security layer (RFC 7465)
- Hacked, deprecated, and not recommended

WPA

- Wi-Fi Protected Access
- 128-bit TKIP (Temporal Key Integrity Protocol) security protocol with RC4 encryption
- Uses 128-bit RC4 (Rivest Cipher 4) encryption algorithm – now prohibited for use in TLS security layer (RFC 7465)
- Interim standard designed to meet security breaches in WEP
- Implements a subset of IEEE 802.11 wireless networking standards – but not all
- Includes a message integrity code (MIC), also known as checksum error detection code, designed to resist digital signature forgery.

WPA2

- Wi-Fi Protected Access 2 - also known as RSN (Robust Secure Network)
- Includes Personal and Enterprise versions
 - [WPA 2 Personal](#) version intended for home use and does not provide the best solution for corporate business environments
 - Lower security but easy to deploy
 - Uses pre-shared keys (WPA2-PSK) with common passcode phrase that can potentially be shared illicitly outside of your corporate organization.
 - [WPA Enterprise](#) is designed primarily for corporate business Wi-Fi.
 - High security but complex to deploy
 - Distributes a unique credential to all network users which eliminates the need for (WPA2-PSK) passcodes.
 - Includes all features of WPA2-Personal with additional support for 802.1x RADIUS server authentication, where remote clients pass user information to databases on RADIUS servers.
 - Uses IEEE 802.1X and the Extensible Authentication Protocol (EAP) that allows authentication with either passwords (not recommended) or certificates (recommended). Authentication is implemented through multiple EAP authentication modes, including [EAP-TLS](#), [TLS-LEAP](#), and [EAP-PEAP](#) methods.
 - User-based access control for up to 32 authenticated clients per port, with port-based network access control (PNAC) option that allows authentication by a single client to open the port. Port-based access does not impose a client limit.
- Mandatory use of 128-bit Advanced Encryption Standard (AES) to encrypt data transmitted over-the-air instead of RC4 (used in earlier WEP/WPA protocols)
- Must use strong CCMP (Counter Mode CBC-MAC Protocol) to encrypt data transmitted over the air - as opposed to TKIP (Temporal Key Integrity Protocol) encryption algorithm previously used in the WEP/WPA protocols that only partly meet IEEE 802.11 wireless networking standard.
- Optional use of protected Management Frames (PMF) provide protection for unicast and multicast management action frames. Unicast management action frames are protected from both eavesdropping and forging, and multicast management action frames are protected from forging.

WPA3

- Next-generation Wi-Fi protocol with new functions to simplify security
- WPA3 has replaced WPA2 but is not widely available. It is possible to upgrade existing systems to support WPA3.
- Mandatory use of protected Management Frames (PMF)
- Like WPA2 it includes both Personal and Enterprise versions:
 - [WPA3 Personal](#) version designed for home use. It implements Simultaneous Authentication of Equals (SAE) to replace the Pre-Shared Key (PSK) exchange protocol (used in WPA2 Personal). It provides forward secrecy and is resistant to offline decryption attacks.
 - Uses 192-bit Advanced Encryption Standard (AES)
 - [WPA3 Enterprise](#) uses IEEE 802.1X and Extensible Authentication Protocol (EAP) and is designed primarily for business use.
 - Based on WPA2-Enterprise and includes new functions with optional 192-bit key length for highly sensitive environments.
 - Supports [EAP-TLS](#) authentication mode only.
 - Uses optional 192-bit Advanced Encryption Standard (AES) – as opposed to 128-bit AES in WPA3 Personal

1.2 Wireless security in u-blox short range modules

The Wi-Fi security modes, including WPA2 (AES-CCMP), WPA/WPA2 Mixed mode (RC4-TKIP + AES-CCMP), WPA (RC4-TKIP), and configuration of pre shared keys (PSK), are configured using the AT command `+UWAPC`. For more information about module security settings, see also the u-connectXpress AT commands manual [3].

Table 1 describes the Wi-Fi security modes and related encryption security protocols supported by u-blox short range standalone modules.

Wi-Fi security mode	Encryption security protocols				
	Unencrypted	TKIP	AES/CCMP	PMF	Key method
Open (no security)	Valid	-	-		
WPA2-PSK (Personal)	-		Valid	Supported	PSK
WPA3-PSK (Personal)	-		Valid	Optional	SAE
802.1X (Enterprise) EAP-TLS, PEAP, LEAP	-	Valid (only for station)	Valid (only for station)		

Table 1: Wi-Fi security support in u-blox short range modules

2 Standards and topologies

2.1 WPA2 Personal

“Personal security” generally refers to WPA2 Personal networks intended for home use. Although this supersedes WPA, this security class offers a lower-level of security as it uses pre-shared keys (WPA2-PSK) with common passcode phase that can potentially be shared illicitly outside of your corporate organization.

Figure 2 shows how the EAPOL (Extensible Authentication Protocol over LAN) is used in the context of WPA2 Personal network that uses pre-shared keys (WPA2-PSK). The protocol provides the encapsulation for the 4-way handshake process used to generate the passkeys for secure communication between the client and Access Point.

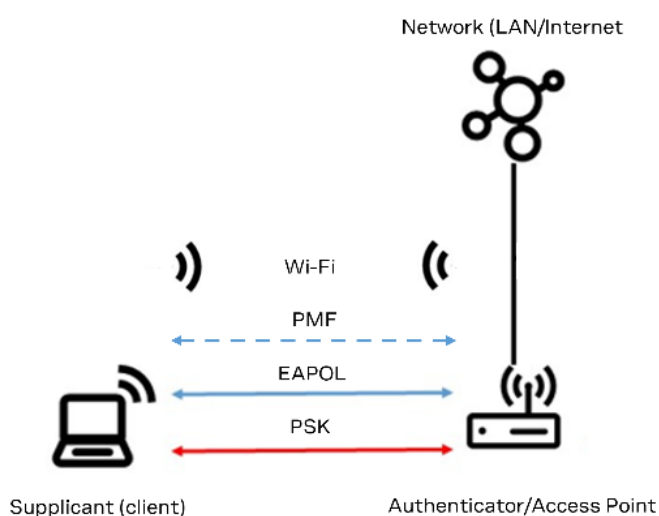


Figure 2: Common WPA2 Personal security topology

Figure 3 shows the network access process between the client and Access Point in WPA2 Personal networks that is based on a 4-way EAPOL handshake between the client and Access Point.

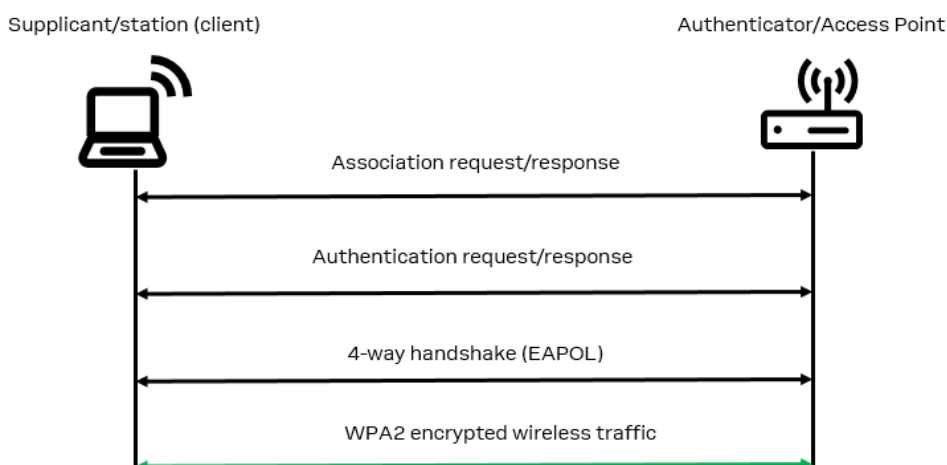


Figure 3: WPA/WPA2 Wi-Fi authentication procedure

Client authentication is performed using a pre-shared key (PSK) derived from a single Wi-Fi password using cryptographic functions. As a result of the authentication process a Pairwise Master Key (PMK) is generated. The PMK is used to encrypt the traffic between the client and access point.

The PSK is verified and PMK is generated in a four-way handshake between the Station and Access Point. Authentication messages during the handshake are carried using EAPOL (Extensible Authentication Protocol over LAN) frames.

Handshakes are encrypted using WPA2 with AES/CCMP. If these more modern protocols are not supported by the access point, WPA with TKIP encryption can be used.

For successful network access of station, the user must provide a correct password and the SSID of the target network.

Figure 4 shows a typical packet exchange between the station ODIN-W2 in this case and access point during WPA2 authentication.

Time	Source	Destination	Protocol	Length	Info
747 2020-04-08 11:20:05,629949	Cisco_24:ed:73	00:00:00_00:00:00	802.11	343	Probe Response, SN=0, FN=0, Flags=....., BI=100, SSID=fae-testnet-1
771 2020-04-08 11:20:06,639964	Cisco_24:ed:73	00:00:00_00:00:00	802.11	343	Probe Response, SN=0, FN=0, Flags=....., BI=100, SSID=fae-testnet-1
791 2020-04-08 11:20:07,649960	Cisco_24:ed:73	00:00:00_00:00:00	802.11	343	Probe Response, SN=0, FN=0, Flags=....., BI=100, SSID=fae-testnet-1
818 2020-04-08 11:20:08,660067	Cisco_24:ed:73	00:00:00_00:00:00	802.11	343	Probe Response, SN=0, FN=0, Flags=....., BI=100, SSID=fae-testnet-1
871 2020-04-08 11:20:10,679958	Cisco_24:ed:73	00:00:00_00:00:00	802.11	343	Probe Response, SN=0, FN=0, Flags=....., BI=100, SSID=fae-testnet-1
887 2020-04-08 11:20:11,689941	Cisco_24:ed:73	00:00:00_00:00:00	802.11	343	Probe Response, SN=0, FN=0, Flags=....., BI=100, SSID=fae-testnet-1
1010 2020-04-08 11:20:18,760047	Cisco_24:ed:73	00:00:00_00:00:00	802.11	343	Probe Response, SN=0, FN=0, Flags=....., BI=100, SSID=fae-testnet-1
1030 2020-04-08 11:20:19,769791	Cisco_24:ed:73	00:00:00_00:00:00	802.11	343	Probe Response, SN=0, FN=0, Flags=....., BI=100, SSID=fae-testnet-1
1066 2020-04-08 11:20:21,789968	Cisco_24:ed:73	00:00:00_00:00:00	802.11	343	Probe Response, SN=0, FN=0, Flags=....., BI=100, SSID=fae-testnet-1
1090 2020-04-08 11:20:22,799953	Cisco_24:ed:73	00:00:00_00:00:00	802.11	343	Probe Response, SN=0, FN=0, Flags=....., BI=100, SSID=fae-testnet-1
1099 2020-04-08 11:20:23,735543	u-blox_82:f7:36	Cisco_24:ed:73	802.11	59	Authentication, SN=1, FN=0, Flags=.....
1100 2020-04-08 11:20:23,737432	Cisco_24:ed:73	u-blox_82:f7:36	802.11	56	Authentication, SN=0, FN=0, Flags=.....
1102 2020-04-08 11:20:23,738388	u-blox_82:f7:36	Cisco_24:ed:73	802.11	144	Association Request, SN=2, FN=0, Flags=....., SSID=fae-testnet-1
1104 2020-04-08 11:20:23,739046	Cisco_24:ed:73	u-blox_82:f7:36	802.11	343	Probe Response, SN=0, FN=0, Flags=....., BI=100, SSID=fae-testnet-1
1105 2020-04-08 11:20:23,739200	Cisco_24:ed:73	u-blox_82:f7:36	802.11	165	Association Response, SN=0, FN=0, Flags=.....
1108 2020-04-08 11:20:23,755768	Cisco_24:ed:73	u-blox_82:f7:36	802.11	48	Action, SN=0, FN=0, Flags=.....
1109 2020-04-08 11:20:23,755872	Cisco_24:ed:73	u-blox_82:f7:36	EAPOL	148	Key (Message 1 of 4)
1112 2020-04-08 11:20:23,756751	u-blox_82:f7:36	Cisco_24:ed:73	802.11	62	Action, SN=4, FN=0, Flags=.....
1113 2020-04-08 11:20:23,758244	u-blox_82:f7:36	Cisco_24:ed:73	EAPOL	184	Key (Message 2 of 4)
1114 2020-04-08 11:20:23,761292	Cisco_24:ed:73	u-blox_82:f7:36	EAPOL	228	Key (Message 3 of 4)
1116 2020-04-08 11:20:23,766762	u-blox_82:f7:36	Cisco_24:ed:73	EAPOL	162	Key (Message 4 of 4)
1128 2020-04-08 11:20:23,809899	Cisco_24:ed:73	00:00:00_00:00:00	802.11	343	Probe Response, SN=0, FN=0, Flags=....., BI=100, SSID=fae-testnet-1
1134 2020-04-08 11:20:23,811581	Cisco_24:ed:73	00:00:00_00:00:00	802.11	343	Probe Response, SN=0, FN=0, Flags=....., BI=100, SSID=fae-testnet-1

Figure 4: Packet exchange during WPA2 authentication

2.2 WPA2 Enterprise (802.1X)

802.1X is standard defined by IEEE, see https://en.wikipedia.org/wiki/IEEE_802.1X.

“Enterprise security” generally refers to WPA2 Enterprise networks secured with 802.1X authentication using a RADIUS server. With network communication based on standard EAP and RADIUS protocols, the server checks the certificates or credentials of users and prevents any unauthorized access to the network.

Figure 5 shows a common 802.1X (WPA2) topology that combines Personal (WPA2) with Enterprise Wi-Fi security. In this scenario, the encapsulation and framing of EAP messages is transported between the Wi-Fi access point (supplicant) and RADIUS server (authenticator).

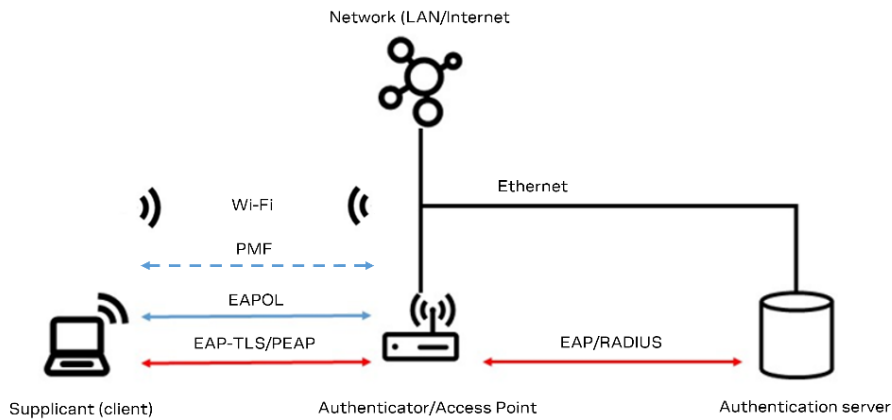


Figure 5: Common 802.1X topology

Figure 6 shows the network access process between the client and Access Point in WPA2 Enterprise networks that includes the 4-way EAPOL handshake.

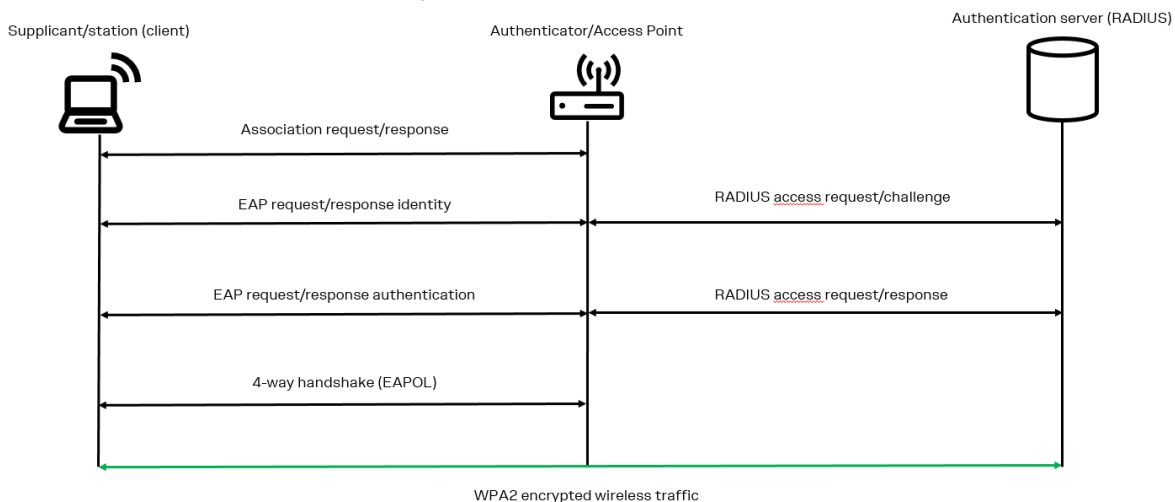


Figure 6: 802.11X EAP authentication procedure

EAP is the underlying protocol for 802.1X and is used for communication between the supplicant (client) and authentication server. EAP provides common message format, and the negotiation process of authentications are called EAP methods. Several methods exist, including [EAP-TLS](#), [LEAP](#) and [PEAP](#). How the authentication is defined and handled depends to the EAP type.

For EAP, the authenticator (Access Point) is only a proxy between the client and authentication server. The authenticator enables and allows the communication.

Although EAP uses certificates and private keys to identify and authenticate clients, certificates are not needed in all situations. For example, PEAP can be used without server authentication or any need to upload the CA certificate. However, server connections that have not been validated are generally considered unsecure and should not be used.

2.3 WPA3 Personal

In WPA3, the WPA2-PSK authentication used in WPA2 personal networks is replaced by WPA3-SAE (Simultaneous Authentication of Equals). The new standard uses 128-bit encryption key and Forward Secrecy protocol to protect against offline dictionary attacks and improve key exchange security.

PMF (Protected Management Frames) is mandated, which means that although it must be supported in WPA3 devices its use is optional.

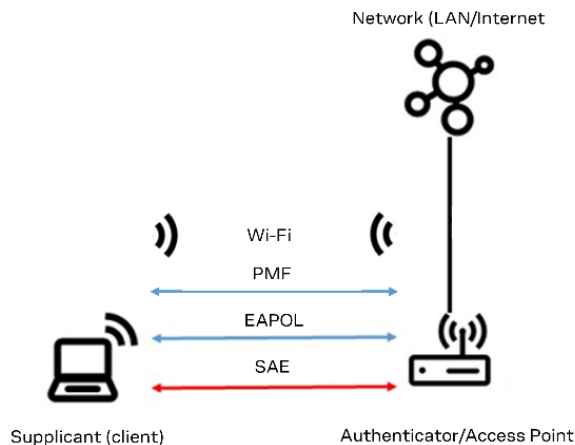


Figure 7: Common 802.1X topology

2.4 WPA3 Enterprise

WPA3-Enterprise is developed based on WPA2-Enterprise and includes new functions that simplify security. Whereas WPA2-Enterprise supports multiple EAP authentication modes, WPA3-Enterprise supports only EAP-TLS authentication. It uses IEEE 802.1X and Extensible Authentication Protocol (EAP).

The 64/128-bit length can be extended with an optional 192-bit security key for even better protection, which can be of particular interest amongst large corporations and other highly sensitive environments. The extended 192-bit security mode can require updates related to the EAP component on the RADIUS server. The HMAC-SHA-384 algorithm is used to export keys in the four-way handshake phase.

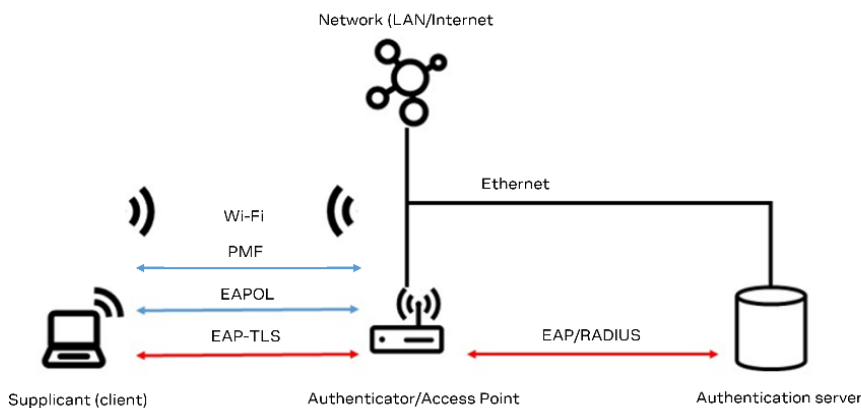


Figure 8: Common 802.1X topology

3 EAP authentication methods

3.1 EAP-TLS

Extensible Authentication Protocol (EAP) is an authentication framework frequently used in network and internet connections. It is defined in RFC 3748.

EAP-TLS is considered to be one of the most secure EAP standards.

See also: https://en.wikipedia.org/wiki/Extensible_Authentication_Protocol

 u-blox short range modules do not support 802.1X authentication in Access Point mode.

 Wi-Fi security and [End-to-End security](#) can be used together.

3.1.1 EAP-TLS usage

Client/User certificates and private keys (optional) are stored in the module and sent to the Server for client validation.

3.1.2 EAP-TLS validation

CA-Root certificates are stored in module and used to validate the Server certificate received during the connection setup.

In some cases, optional parameters, like User Name, Password and Domain, are used to identify users and strengthen security.


3.2 PEAP

The Protected Extensible Authentication Protocol (PEAP), also known as Protected EAP, encapsulates the Extensible Authentication Protocol (EAP) within an encrypted and authenticated Transport Layer Security (TLS) tunnel. PEAP is considered as a secure protocol and is widely used.

See also https://en.wikipedia.org/wiki/Protected_Extensible_Authentication_Protocol

3.2.1.1 PEAP usage

When using PEAP as authentication protocol the Username, Password and Domain (optional) are used to identify client and together with the CA-certificate on the module. As with EAP-TLS, the CA-certificate stored in module is used to validate the server certificate received during the connection setup.

 PEAPv0 (with EAP-MSCHAPv2) is supported by ODIN-W2 and NINA-W13/15.

 PEAPv1 (with EAP-GTC) is not supported by ODIN-W2 and NINA-W13/15.

3.2.1.2 PEAP server validation

The server validation is an important feature in PEAP and should not be disabled.

If server validation is not enabled on the module, it is possible to intercept and decrypt client credentials like Username and Password.

3.3 LEAP

Lightweight Extensible Authentication Protocol (LEAP).


With LEAP no certificates (client-side or server-side) are required. Authentication is based on dynamic WEP keys and mutual authentication between the client and server.

LEAP offers moderately high security, but only when strong passwords are used.

For more information: https://en.wikipedia.org/wiki/Lightweight_Extensible_Authentication_Protocol

4 TLS end-to-end security

Transport Layer Security (TLS) and its now deprecated predecessor, Secure Sockets Layer (SSL), are cryptographic protocols designed for secure end-to-end communications security over a computer network.

 Wi-Fi security and End-to-End security can be used together.

End-to-End Transport Layer Security – for internet security over TCP connections

- Uses TLS (Transport Layer Security)
- Used for example for HTTPS and MQTT

For more information see also: https://en.wikipedia.org/wiki/Transport_Layer_Security

4.1 TLS over TCP

For secure connection to web pages and the cloud it is very important that the data is secured. Secure communication protocols like HTTPS and MQTT protect with encrypt data using TLS and are popular on the internet.

4.2 TLS over TCP usage

To connect using TLS over TCP, the CA-Root certificate is stored in the module and during the TCP connection. The TLS is set up if the server accepts the supported cipher suites and the CA-Root validation is successful. It is also possible to connect to TLS without server validation, but then the connection must be considered less secure.

4.3 TLS features supported by u-blox

U-blox short range modules use mbed TLS. For more information: <https://tls.mbed.org/>.

4.3.1 Supported TLS versions and features

Table 2 shows the TLS versions that are supported in ODIN-W2 and NINA-W13/15 modules.

ODIN-W2		NINA-W13/15		Supported versions
Wi-Fi	TCP	Wi-Fi	TCP	
Yes	Yes	Yes	Yes	TLS 1.0
Yes	Yes	No	Yes	TLS 1.1
Yes	Yes	No	Yes	TLS 1.2
No	No	No	No	TLS 1.3

Table 2: TLS versions supported

Table 3 shows the TLS extensions that are supported in ODIN-W2 and NINA-W13/15 modules.

ODIN-W2		NINA-W13/15		TLS Supported extensions
Wi-Fi	TCP	Wi-Fi	TCP	
No	Yes	No	Yes	Server Name Indication (SNI)
No	Yes	No	Yes	Truncated HMAC

Table 3: TLS extensions supported

4.3.2 TLS cipher suites supported by ODIN-W2

TLS_RSA_WITH_3DES_EDE_CBC_SHA (0x000a)
TLS_RSA_WITH_RC4_128_SHA (0x0005)
TLS_RSA_WITH_RC4_128_MD5 (0x0004)

TLS_RSA_WITH_AES_128_CCM (0xc09c)
TLS_RSA_WITH_AES_256_CCM (0xc09d)
TLS_RSA_WITH_AES_256_CBC_SHA256 (0x003d)
TLS_RSA_WITH_AES_256_CCM_8 (0xc0a1)
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (0xc027)
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
TLS_RSA_WITH_AES_128_GCM_SHA256 (0x009c)
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 (0xc023)
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b)
TLS_RSA_WITH_AES_128_CBC_SHA (0x002f)
TLS_RSA_WITH_AES_256_CBC_SHA (0x0035)
TLS_RSA_WITH_AES_128_CBC_SHA256 (0x003c)
TLS_EMPTY_RENEGOTIATION_INFO_SCSV (0x00ff)

Table 4: TLS cipher suites supported by ODIN-W2 version 8.0.0 or newer

4.3.3 TLS cipher suites supported by NINA-W13/15

TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc02c)
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 (0x009f)
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 (0xc024)
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (0xc028)
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 (0x006b)
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (0xc00a)
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)
TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x0039)
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b)
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 (0x009e)
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 (0xc023)
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (0xc027)
TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 (0x0067)
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (0xc009)
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)
TLS_DHE_RSA_WITH_AES_128_CBC_SHA (0x0033)
TLS_RSA_WITH_AES_256_GCM_SHA384 (0x009d)
TLS_RSA_WITH_AES_256_CBC_SHA256 (0x003d)
TLS_RSA_WITH_AES_256_CBC_SHA (0x0035)
TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384 (0xc032)
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384 (0xc02a)
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA (0xc00f)
TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384 (0xc02e)
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384 (0xc026)
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA (0xc005)
TLS_RSA_WITH_AES_128_GCM_SHA256 (0x009c)
TLS_RSA_WITH_AES_128_CBC_SHA256 (0x003c)
TLS_RSA_WITH_AES_128_CBC_SHA (0x002f)
TLS_ECDH_RSA_WITH_AES_128_GCM_SHA256 (0xc031)
TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256 (0xc029)
TLS_ECDH_RSA_WITH_AES_128_CBC_SHA (0xc00e)
TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02d)
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256 (0xc025)
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA (0xc004)
TLS_EMPTY_RENEGOTIATION_INFO_SCSV (0x00ff)

Table 5: TLS Cipher suites supported by NINA-W13/15

4.3.4 TLS groups supported by ODIN-W2 and NINA-W13/15

secp521r1 (0x0019)
brainpoolP512r1 (0x001c)
secp384r1 (0x0018)
brainpoolP384r1 (0x001b)
secp256r1 (0x0017)
secp256k1 (0x0016)
brainpoolP256r1 (0x001a)
secp224r1 (0x0015)
secp224k1 (0x0014)
secp192r1 (0x0013)
secp192k1 (0x0012)

Table 6: TLS groups (Elliptic curves) supported by ODIN-W2 and NINA-W13/15

5 Certificates and private keys


Certificate formats are based on the ITU-T X.509 recommendation.


OpenSSL is a cryptography library and SSL/TLS toolkit used to generate certificates and keys. It is particularly useful for converting certificate types, viewing and verifying certificates.

You use the toolkit to access relevant file locations and access other user information derived automatically from the configuration file (`openssl.cfg`).

The configuration file resides in a different location depending on the host operating system. On Microsoft Windows it is installed in the folder `C:\Program Files\OpenSSL-Win64\bin`. The configuration can also be given as input when certificates are created.

For more information: <https://www.openssl.org/>

 Certificates normally contain additional metadata that increases the file size of the certificate. The metadata is related to data fields like issuer, subject, or any other field containing data.

 Data is given as input by the user, when certificates and keys are generated.

5.1 Certificates and encodings

X.509 certificates are digital documents that have been encoded and/or signed in accordance with RFC 5280. The term X.509 certificate usually refers to IETF's PKIX Certificate and CRL Profile of the X.509 v3 certificate standard, as specified in RFC 5280.

An X.509 certificate contains the public key and identity of the party requesting authentication. The certificate is signed when it is generated.

The certificate can be either self-signed, signed by private certificate authority CA, or signed by trusted CA. Self-signed certificates are usually used for testing purposes, and not in production networks. A trusted CA is a public (usually commercial) certificate issuer like Symantec and DigiCert.

A Certificate Signing Request (CSR) is a formal request for the CA to sign, generate, and issue the certificate. CSRs generate a (private and public) key pair and are signed with a secret private key. The private key should never be revealed to outsiders and must be kept secret.

CSRs contain organizational information, such as the domain and optional password.

Private Enhanced Mail (PEM) containers, originally designed for secure email, include public and private keys and can be encrypted. These containers do not support certificate encryption.

PEM files can contain more than one certificate and are subsequently sent as "chained" certificates.

5.1.1 X.509 file extensions

There is often some confusion about DER, PEM, CRT, and CER file extensions. Although these can be interchanged in some cases, these file extensions are generally not interchangeable. Consequently, it is good practice to identify how your certificate is encoded and then label it correctly.

PEM (Privacy-Enhanced Mail) format

PEM formatted keys can be encrypted, but certificate encryption is not supported.

PEM files can contain more than one certificate. If so, then the certificates are said to be "chained" and will be sent as certificate chain.

DER (Distinguished Encoding Rules) format

Although u-blox short range modules accept certificates and keys in both PEM and DER formats, PEM files are converted to DER format by the module and stored to internal memory.

u-blox standalone modules support certificates with 4 kB key length up to 8 kB in size.

CRT

5.2 Operations with certificates and keys

5.2.1 How to create certificates and keys

5.2.1.1 Create CA certificate example

```
openssl req -config openssl.cnf -newkey rsa:4096 -keyout ca.key.pem -nodes -new -x509 -
days 7200 -sha256 -extensions v3_ca -out ca.cert.pem -subj
"/C=CH/ST=Zuerich/L=Thalwil/O=ublox AG/OU=fae-emea/CN=ublox Wireless CA/emailAddress=root-
ca@u-blox.com"
```

5.2.1.2 Create client CSR and certificate example

Create CSR:

```
openssl req -config openssl.cnf -newkey rsa:2048 -keyout client.key.pem -nodes -new -
sha256 -out client.csr.pem -subj "/C=CH/ST=Zuerich/L=Thalwil/O=ublox AG/OU=fae-
emea/CN=ublox Wireless User/emailAddress=user@u-blox.com"
```

Create certificate:

```
openssl ca -config openssl.cnf -extensions user_cert -days 365 -notext -md sha256 -in
client.csr.pem -out client.cert.pem
```

5.2.1.3 Create server CSR and certificate example

Create CSR:

```
openssl req -config openssl.cnf -newkey rsa:2048 -keyout server.key.pem -nodes -new -
sha256 -out server.csr.pem -subj "/C=CH/ST=Zuerich/L=Thalwil/O=ublox AG/OU=fae-
emea/CN=ublox Wireless Server/emailAddress=server@u-blox.com"
```

Create certificate:

```
openssl ca -config openssl.cnf -extensions server_cert -days 365 -notext -md sha256 -in
server.csr.pem -out server.cert.pem
```

5.2.2 How to view certificates

Although PEM encoded certificates are created in ASCII, they are not in a natively human-readable format and must normally be converted.

For example, use the following `openssl x509` command to make the `server.cert.pem` certificate created in the previous section readable.

```
openssl x509 -in server.cert.pem -text -noout
```

Certificate:

Data:

```
Version: 3 (0x2)
Serial Number: 1 (0x1)
Signature Algorithm: sha256WithRSAEncryption
```

Issuer: C = CH, ST = Zuerich, L = Thalwil, O = ublox AG, OU = fae-emea, CN = ublox Wireless CA, emailAddress = root-ca@u-blox.com

Validity

Not Before: Mar 30 09:18:20 2020 GMT

Not After : Mar 30 09:18:20 2021 GMT

Subject: C = CH, ST = Zuerich, L = Thalwil, O = ublox AG, OU = fae-emea, CN = ublox Wireless Server, emailAddress = server@u-blox.com

Subject Public Key Info:

Public Key Algorithm: rsaEncryption

RSA Public-Key: (2048 bit)

Modulus:

00:db:35:97:a3:58:be:0c:15:c8:e8:12:ca:0a:9e:
2c:2e:3a:b0:85:7d:df:27:a2:8d:7a:16:7b:06:92:
93:31:f6:82:f7:7f:39:66:3f:0c:03:22:e0:59:5d:
72:b3:e4:90:cb:56:25:f9:3e:61:15:fa:9c:74:25:
66:ac:cc:4e:d3:d6:5b:7d:e0:ef:dd:4b:3b:66:61:
7e:c7:19:cf:4e:af:18:a0:73:b9:82:94:17:bb:7a:
e4:85:24:1e:61:1e:9e:65:82:3c:52:21:12:a1:f7:
82:a4:44:f5:b9:da:fc:38:73:f6:04:5e:16:4b:34:
b7:b9:ce:1c:b3:01:d7:20:25:b1:db:07:be:22:32:
c6:54:93:db:93:91:74:55:49:b5:ff:fd:ea:f9:da:
24:bf:b3:dc:e1:31:80:a8:5e:d3:25:3f:3c:f3:e3:
d1:e6:a1:e3:ba:5b:37:45:6b:08:3c:b9:e7:3a:7a:
8c:f5:b6:db:1a:2c:32:56:26:96:d4:6e:95:98:3e:
02:b6:c3:09:8a:56:d3:8e:48:e3:17:36:d9:32:37:
0c:ae:29:a7:37:bb:e9:e0:62:3b:f7:f3:88:fb:ea:
fc:66:9b:ff:85:6a:41:14:c3:f7:cc:e2:e4:0c:70:
8b:5b:39:a6:4a:23:f9:4d:02:ae:26:d6:b3:a3:e1:
c4:8f

Exponent: 65537 (0x10001)

X509v3 extensions:

X509v3 Basic Constraints:

CA:FALSE

Netscape Cert Type:

SSL Server

Netscape Comment:

OpenSSL Generated Server Certificate

X509v3 Subject Key Identifier:

C4:C7:29:D8:E2:AA:5B:A5:6C:11:02:07:56:CB:41:85:83:5B:55:80

X509v3 Authority Key Identifier:

keyid:74:D5:7F:0B:12:A3:10:26:13:04:95:84:43:D9:71:45:31:30:89:86

DirName:/C=CH/ST=Zuerich/L=Thalwil/O=ublox AG/OU=fae-emea/CN=ublox

Wireless CA/emailAddress=root-ca@u-blox.com

serial:0C:A5:47:06:A7:2A:EF:5A:2A:54:9A:1D:A8:DC:14:F5:D6:DA:70:95

X509v3 Key Usage: critical

Digital Signature, Key Encipherment

X509v3 Extended Key Usage:

TLS Web Server Authentication

Signature Algorithm: sha256WithRSAEncryption

9b:48:d7:67:9a:94:36:c7:e9:45:a7:a1:7f:85:be:48:89:1d:
81:0c:44:17:0d:54:c3:33:8c:0a:a7:88:c0:23:bc:b4:2c:72:
f2:9d:40:e1:ca:11:a5:1d:53:03:ff:00:a8:f3:79:7b:0e:1f:
89:73:50:86:02:7a:af:90:55:bf:43:18:1f:ad:f3:2c:b1:e1:
bc:af:79:62:25:29:6f:49:03:12:e3:4b:3a:2c:76:7a:17:1b:
b0:bf:10:78:9a:04:fe:44:33:ba:a2:73:57:65:9f:16:83:82:
0f:96:ab:a8:d2:c7:16:bd:cf:7c:2c:12:e7:ff:79:81:a4:73:
84:c8:df:bf:a5:38:9c:eb:c7:16:0b:82:59:6b:59:5a:03:87:
06:bc:c2:c6:5c:95:95:98:1b:f4:3f:c1:55:64:a2:8c:a8:00:
94:33:0f:a5:f1:1d:e5:b5:8d:78:64:f4:b2:c4:ab:fe:a7:c0:
f9:9f:83:dd:46:0f:71:c0:30:53:bc:58:5f:72:7e:bd:a0:6a:
fc:f3:5c:87:fa:f1:41:27:6f:e3:8a:7e:b7:36:0f:85:43:e2:
82:50:66:a8:29:0e:02:d5:83:a9:d7:10:76:09:00:88:10:a5:
ab:06:8a:4c:4b:6a:e9:46:68:2f:dc:82:96:a8:80:9a:69:68:
e9:fb:dd:4e:a0:20:bd:68:6d:f1:d7:47:c8:23:16:db:fc:64:
d4:2d:8f:8e:ff:a9:a9:07:00:f8:0f:e9:96:f2:21:ed:a4:dc:
7a:6f:a0:f9:0a:a3:d2:23:1b:f5:6d:2e:fe:39:ba:0b:2f:45:
77:15:c2:f5:45:f5:44:7f:ef:64:89:64:fc:05:72:9e:6e:76:

```
44:42:98:c8:ea:35:b0:99:c4:78:54:b7:40:21:35:f5:7a:6e:
b0:24:4c:2a:ed:61:81:88:d3:b0:cc:da:b3:9c:7e:76:08:4b:
99:73:6d:1f:1c:2d:86:0f:77:0f:c9:e8:cc:7b:5a:2b:7c:0c:
62:62:f7:7e:9d:16:eb:39:d2:c5:c7:e6:ee:e1:e3:ab:f3:0c:
39:fa:35:00:19:17:65:03:49:7c:e8:67:09:44:77:76:b2:91:
a0:18:cd:8d:35:34:e0:ca:64:f4:cb:fc:1d:f8:01:d0:06:7c:
b7:d3:e4:56:94:7e:ac:ad:a0:79:f9:85:07:89:c8:39:8b:94:
4b:e0:9f:66:99:36:85:de:df:c7:ff:e3:a2:2a:67:22:47:aa:
e7:e1:07:df:07:b2:67:bb:6f:85:59:26:cd:1d:99:4f:fd:24:
1a:ca:d0:e6:28:1e:eb:4c:20:c7:8e:a9:7e:38:8e:22:2d:d5:
f2:7c:90:ff:7c:21:1f:65
```

5.2.3 How to convert certificate types


Certificates can be converted from one type of encoded certificate to another. The commands for converting certificates between PEM and DER format are shown below.

PEM to DER:

```
openssl x509 -in cert.crt -outform der -out cert.der
```

DER to PEM:

```
openssl x509 -in cert.crt -inform der -outform pem -out cert.pem
```

 In some cases, it is advantageous to combine multiple pieces of the X.509 infrastructure into a single file. One common example would be to combine both the private key and public key into the same certificate.

6 Configuration examples

6.1 Personal security

6.1.1 WPA2

The following configuration stores the settings in the flash memory of the module. The module automatically tries to connect to network on power up.

1. Access point has been configured with network parameters (SSID, password)

```
# Set Wi-Fi to be active at startup.
AT+UWSC=0,0,1
# Set SSID for the Network.
AT+UWSC=0,2,"UBXWifi"
# Use WPA2 as authentication type.
AT+UWSC=0,5,2
# Use Password "my_password".
AT+UWSC=0,8,"my_password"
# Store the Wi-Fi Station configuration.
AT+UWSCA=0,1
# Store configuration to the startup database.
AT&W
# Reboot the u-blox short range module.
AT+CPWROFF
```

6.2 Enterprise security 802.11X

6.2.1 Example RADIUS server setup

In this example FreeRADIUS is used to setup EAP-TLS ready authentication. Default file locations are used on RADIUS server.

For more information about FreeRADIUS, visit: <https://freeradius.org/>

1. Set file paths and folders on the RADIUS general configuration.

```
sudo nano /etc/freeradius/3.0/radiusd.conf
```

For example:

```
prefix = /usr
exec_prefix = /usr
sysconfdir = /etc
localstatedir = /var
sbindir = ${exec_prefix}/sbin
# RADIUS logs collect to this directory.
logdir = /var/log/freeradius
raddbdir = /etc/freeradius/3.0
# Directory for account data.
radacctdir = ${logdir}/radacct

#
# name of the running server. See also the "-n" command-line option.
name = freeradius

# Location of config and logfiles.
confdir = ${raddbdir}
modconfdir = ${confdir}/mods-config
# Location of Root-CA.
cadir = /home/pi/root-ca
# Location of certificates and keys.
certdir = ${cadir}/certs
run_dir = ${localstatedir}/run/${name}
```

2. Configure TLS and certificates information on RADIUS server.

```
sudo nano /etc/freeradius/3.0/mods-available/eap
```

This file is used to set, for example, the EAP type and expected file locations for certificates and keys.

When using EAP-TLS methods, a server-side certificate is needed. Note, that this might change depending on the EAP method.

Server certificates can be self-signed, or signed by private CA or trusted CA. Self-signed certificates are good for testing purposes. Several examples of self-signed certificates are shown in [Enterprise security \(WPA2 Enterprise - 802.1X\)](#).

File locations are set on the tls-common section. For example:

```
tls-config tls-common {
    private_key_password = my_password_802_1X
    private_key_file = ${cadir}/private/server.key.pem
    certificate_file = ${certdir}/server.cert.pem
    ca_file = ${cadir}/certs/ca.cert.pem
}
```

3. Configure Wi-Fi Access Point (AP) information on RADIUS server.

```
sudo nano /etc/freeradius/3.0/clients.conf
```

Based on this information the AP can access the server. For example, the information will give access for AP with IP address 192.168.1.50. 'secret' must be set accordingly on the AP.

```
GNU nano 3.2 /etc/freeradius/3.0/clients.conf

client localhost {
    ipaddr = 127.0.0.1
    proto = *
    secret = testing123
    nas_type = other # localhost isn't usually a NAS...
}

client cisco_wap371_ap {
    ipaddr = 192.168.1.50
    secret = testing123
    proto = *
    nas_type = cisco
}
```

4. Upload certificates and keys to the server. Note the file locations set on point #2.

For example, FileZilla can be used for upload.

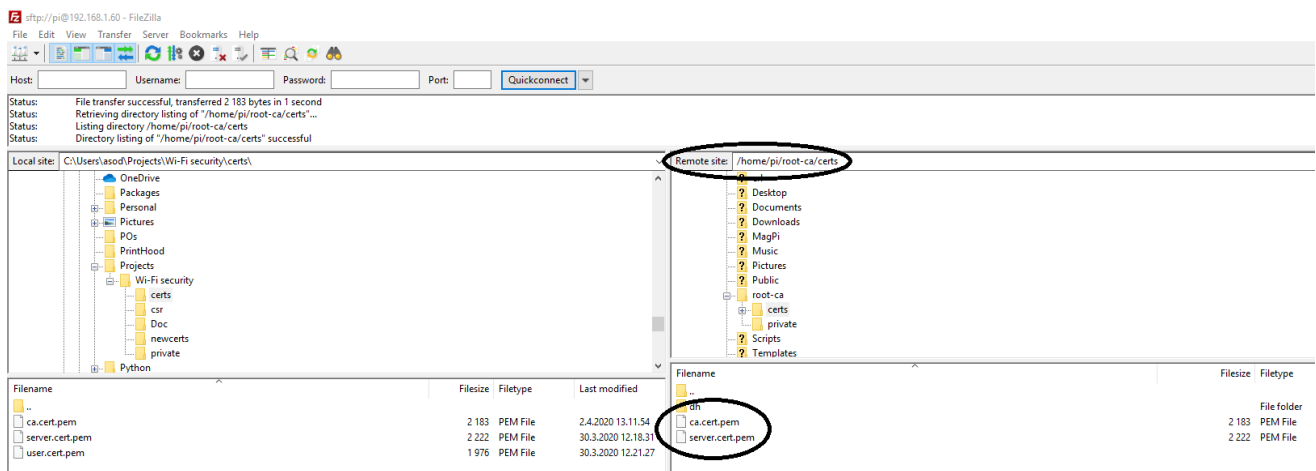


Figure 9: CA certificate and server certificate uploaded to server

5. Restart server. After this server is ready to process authentication request.

```
sudo service freeradius restart
```

The server fails to start if any the configuration is incorrect or files are missing.

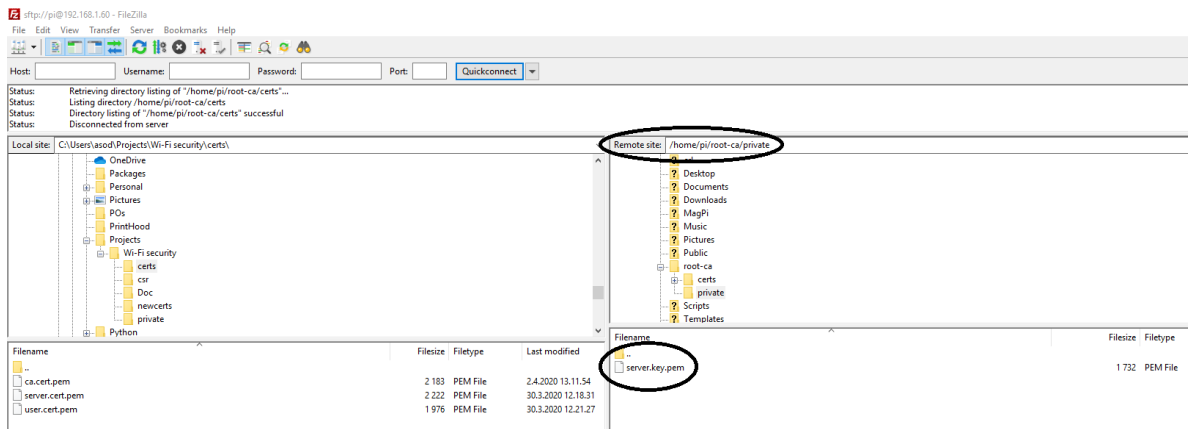


Figure 10: Server key uploaded to server

6.2.2 How to upload certificates and keys to short range module

Note, certificate and/or key must be valid. Otherwise import will fail.

6.2.2.1 Using console application

In this example TeraTerm is used.

1. Open console application and connect to the module.
2. Open certificate or key file with text editor (for example Notepad++). Make note of the file length and check that the encoding is Unix.

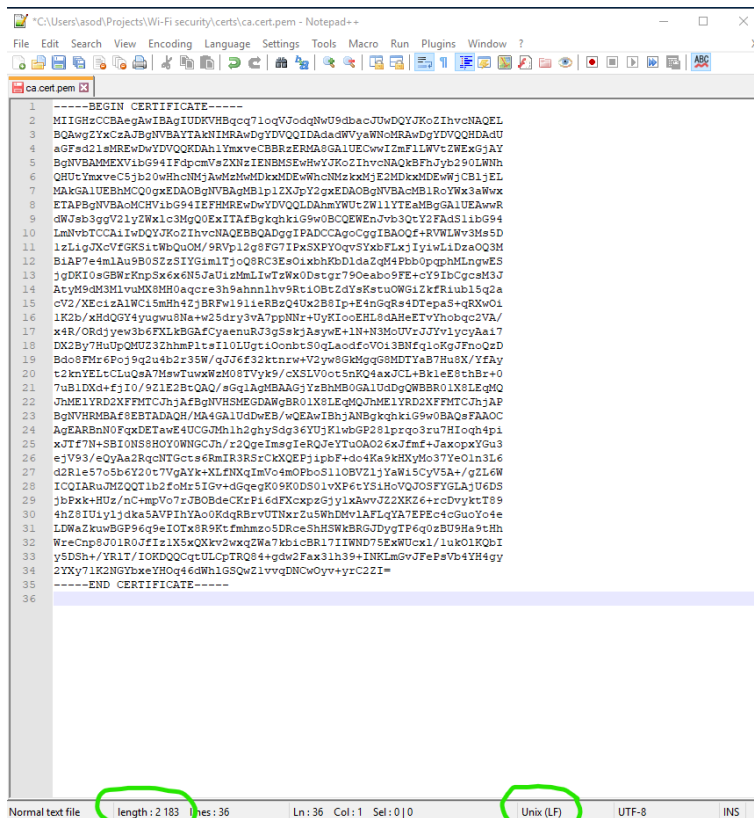


Figure 11: Viewing the certificate in Notepad++

3. Type the command: 'AT+USECMNG=0,<type of the data>,<internal name for the data>,<amount of data>'.
4. When '>' is received, copy the certificate/key contents from the text editor and paste to console window. Press 'enter'.

```
AT+USECMNG=0,0,"rootCA",2183
>
+USECMNG:0,0,"rootCA","7BED8A8A2B49F0944F7022BFDEFCE18B"
OK
```

6.2.2.2 Using s-center

1. Open s-center and navigate to "Advanced Connection And Settings/Wi-Fi Certificate".
2. Select the certificate or key type from the drop-down list.
3. Locate the certificate or key file from your PC.
4. Select "Upload".
5. Repeat the above steps for all required files.
6. After all required files are imported, click "Use Certificate".

Figure 12 shows successful import of all three files (CA certificate, client certificate, client private key).

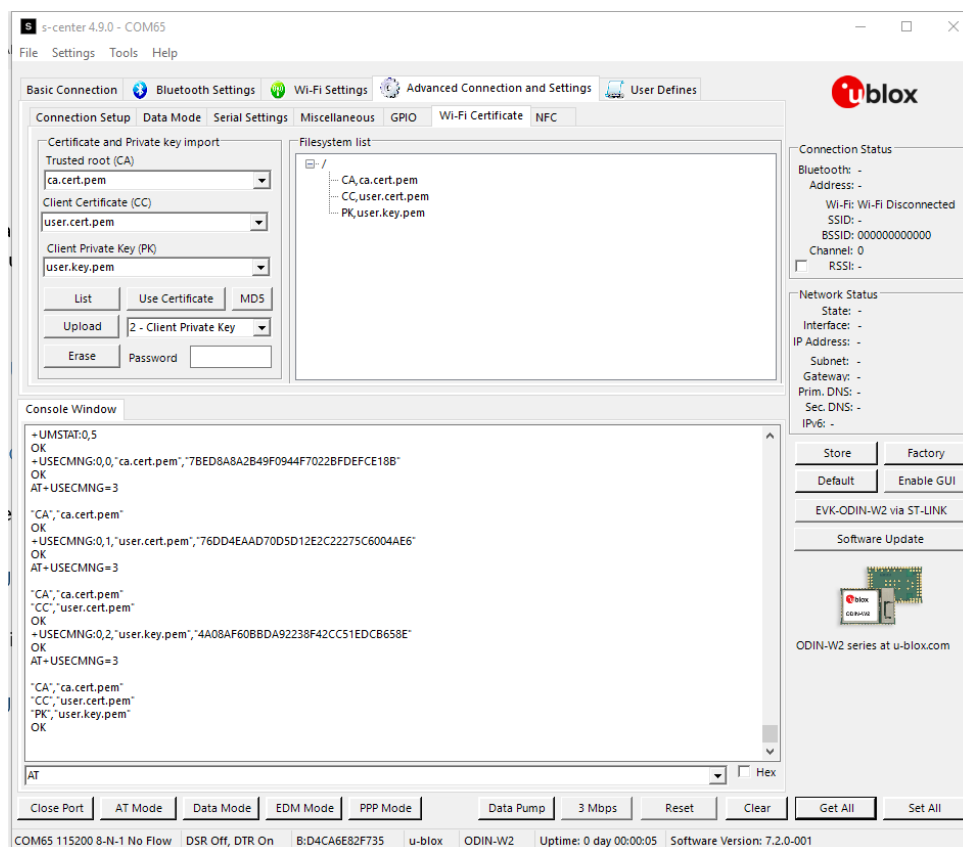


Figure 12: Uploading certificates using s-center

6.2.3 PEAP

The following configuration will store the settings in the flash memory of the module.

On power up the module will start in command mode and automatically try to connect to network.

The following example uses the use of the certificates and keys described in the section 2.2.

1. Configure network and security parameters on Access Point.
2. Configure RADIUS server and import required certificates.
3. Import CA certificate to u-blox short range module. See examples on the section 3.2.1.
4. Configure u-blox short range module.

```
# Set Wi-Fi to be active at startup.
AT+UWSC=0,0,1
# Set SSID for the Network.
AT+UWSC=0,2,"UBXWifi_802_1X"
# Use PEAP as authentication type.
AT+UWSC=0,5,4
# Use Password "my_password_802_1X".
AT+UWSC=0,9,"my_password_802_1X"
# Use user name "my_username_802_1X".
AT+UWSC=0,10,"my_username_802_1X"
# Use the uploaded certificates and keys.
AT+UWSC=0,14,"ca.cert.pem"
# Use server validation for CA certificate. Optional setting, but highly recommended.
AT+UWSC=0,15,1
# Store the Wi-Fi Station configuration.
AT+UWSCA=0,1
# Store configuration to the startup database.
AT&W
# Reboot the u-blox short range module.
AT+CPWROFF
```

6.2.4 LEAP

The following configuration stores the settings in the flash memory of the module.

On power up the module starts in command mode and automatically tries to connect to the network.

The following example uses the use of the certificates and keys described in the section 2.2.

1. Configure network and security parameters on Access Point.
2. Configure RADIUS server and import required certificates.
3. Import CA certificate to u-blox short range module. See examples on the section 3.2.1.
4. Configure u-blox short range module.

```
# Set Wi-Fi to be active at startup.
AT+UWSC=0,0,1
# Set SSID for the Network.
AT+UWSC=0,2,"UBXWifi_802_1X"
# Use LEAP as authentication type.
AT+UWSC=0,5,3
# Use Password "my_password_802_1X".
AT+UWSC=0,9,"my_password_802_1X"
# Use user name "my_username_802_1X".
AT+UWSC=0,10,"my_username_802_1X"
# Use the uploaded certificates and keys.
AT+UWSC=0,14,"ca.cert.pem"
# Use server validation. Optional setting, but highly recommended.
AT+UWSC=0,15,1
# Store the Wi-Fi Station configuration.
AT+UWSCA=0,1
# Store configuration to the startup database.
AT&W
# Reboot the u-blox short range module.
AT+CPWROFF
```

6.2.5 EAP-TLS

The following configuration will store the settings in the flash memory of the module.

On power up the module will start in command mode and automatically try to connect to network.

The following example uses the use of the certificates and keys described in the section 2.2.

1. Configure the network and security parameters on Access Point.
2. Configure RADIUS server and import required certificates.
3. Import certificates and keys to u-blox short range module. See examples on the section 3.2.1.
4. Configure u-blox short range module.

```
# Set Wi-Fi to be active at startup.
AT+UWSC=0,0,1
# Set SSID for the Network.
AT+UWSC=0,2,"UBXWifi_802_1X"
# Use EAP-TLS as authentication type.
AT+UWSC=0,5,5
# Use Password "my_password_802_1X".
AT+UWSC=0,9,"my_password_802_1X"
# Use user name "my_username_802_1X".
AT+UWSC=0,10,"my_username_802_1X"
# Use the uploaded certificates and keys.
AT+UWSC=0,12,"user.cert.pem"
AT+UWSC=0,13,"user.key.pem"
AT+UWSC=0,14,"ca.cert.pem"
# Store the Wi-Fi Station configuration.
AT+UWSCA=0,1
# Store configuration to the startup database.
AT&W
# Reboot the u-blox short range module.
AT+CPWROFF
```

6.3 End-to-end security

To connect using end-to-end security over TCP, it is popular to use an HTTPS connection using TLS to encrypt the link.

1. Download the CA root certificate from the website, in this example www.google.com.
Google provides their CA root certificates on this site <https://pki.goog/>

CA certificates

Root CAs

Name	Public Key	Fingerprint (SHA1)	Valid Until	Links	Tests
GTS Root R1	RSA 4096	e1:c9:50:e6:ef:22:f8:4c:56:45:72:8b:92:20:60:d7:d5:a7:a3:e8	Jun 22, 2036	DER CRL	g r e
GTS Root R2	RSA 4096	d2:73:96:2a:2a:5e:39:9f:73:3f:e1:c7:1e:64:3f:03:38:34:fc:4d	Jun 22, 2036	DER CRL	g r e
GTS Root R3	ECC 384	30:d4:24:6f:07:ff:db:91:89:8a:0b:e9:49:66:11:eb:8c:5e:46:e5	Jun 22, 2036	DER CRL	g r e
GTS Root R4	ECC 384	2a:1d:60:27:d9:4a:b1:0a:1c:4d:91:5c:cd:33:a0:cb:3e:2d:54:cb	Jun 22, 2036	DER CRL	g r e
GS Root R2	RSA 2048	75:e0:ab:b6:13:85:12:27:1c:04:f8:5f:dd:de:38:e4:b7:24:2e:fe	Dec 15, 2021	DER CRL	g r e
GS Root R4	ECC 256	69:69:56:2e:40:80:f4:24:a1:e7:19:9f:14:ba:f3:ee:58:ab:6a:bb	Jan 19, 2038	DER CRL	g r e

Figure 13: Google public root certificates

2. Download the **GS Root R2 RSA 2048**, select the DER format and the **GSR2.crt** file.

```
AT+USECMNG=0,0,GSR2.crt,958
>
+USECMNG:0,0,"GSR2.crt","9414777E3E5EFD8F30BD41B0CFE7D030"
OK
```

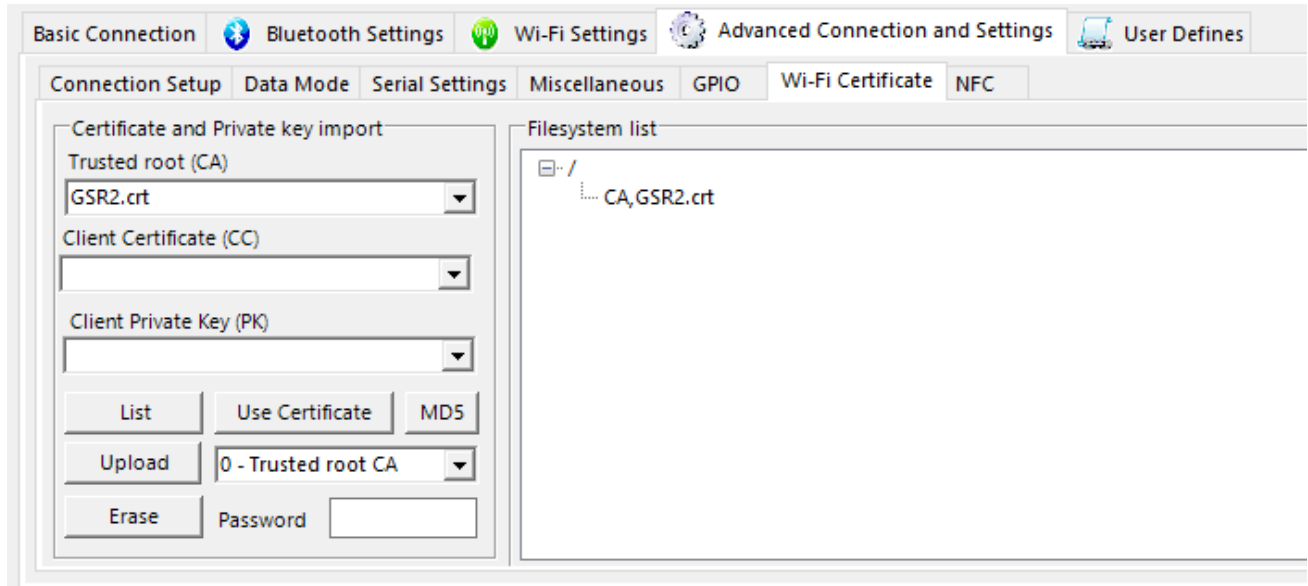


Figure 14: Upload of CA root certificate

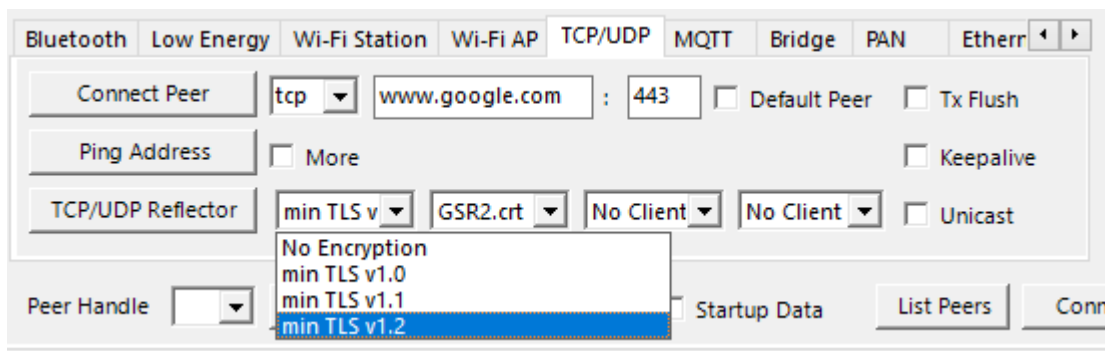


Figure 15: Select the minimum TLS version to be used

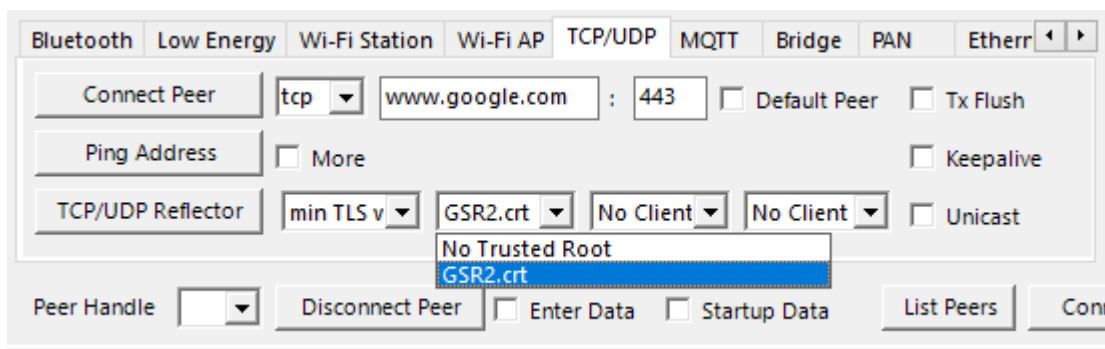


Figure 16: Select the CA root certificate

3. Connect to a Wi-Fi network.
4. Enter the host www.google.com and the port 443 for HTTPS connections.

5. Now connect using s-center or type:

```
AT+UDCP="tcp://www.google.com:443/?encr=3&ca=GSR2.crt"
```

```
+UDCP:1
```

```
OK
```

```
+UUDDPC:1,2,0,10.12.71.57,56448,172.217.168.4,443
```

6. Enter Data mode by ATO1.

7. To download the index file use the following (include the two empty lines):

```
GET /index.html HTTP/1.1
```

```
Connection: keep-alive
```

```
Host: www.google.com
```

```
\r\n
```

```
\r\n
```

```
47 45 54 20 2F 69 6E 64 65 78 2E 68 74 6D 6C 20
48 54 54 50 2F 31 2E 31 0D 0A 43 6F 6E 6E 65 63
74 69 6F 6E 3A 20 6B 65 65 70 2D 61 6C 69 76 65
0D 0A 48 6F 73 74 3A 20 77 77 77 2E 67 6F 6F 67
6C 65 2E 63 6F 6D 0D 0A 0D 0A
```

```
GET /index.html
HTTP/1.1..Connec
tion: keep-alive
..Host: www.goog
le.com....
```

7 Security recommendations

7.1 Personal security

Customers are strongly advised to use WPA2 security or WPA3 for securing Wi-Fi connections. WPA includes some features from older standards, for backward compatibility, and is consequently weaker in some specific cases.

7.2 Enterprise security 802.11X

Customers are advised to upload a trusted Root Certificate Authority (`ca.cert.pem`) file and enable server validation on the device. Server validation is recommended for PEAP, LEAP and EAP-TLS modes. The Wi-Fi device is at risk to variety of Man-In-The-Middle (MITM) attacks without the appropriate proper server validation. An MITM attack can be initiated by any attacker within the Wi-Fi range of the device.

7.3 TLS end-to-end security

7.3.1 TLS for cloud applications

IETF deprecated TLS versions 1.0 and 1.1 as of the 25th of March 2021. u-blox are soon to withdraw support for the deprecated protocol versions on its Wi-Fi products. Consequently, customers are strongly advised to use only TLS version 1.2 for End-to-end security for cloud applications.

7.3.2 Server authentication

Customers are strongly advised to upload a trusted Root Certificate Authority (`ca.cert.pem`) file and enable server authentication on the device. Without proper server validation, the Wi-Fi device is at risk to a variety of MITM attacks. An MITM attack can be initiated by any attacker in the network.

7.3.3 Mutual authentication

Customers are advised to turn on mutual authentication while enabling end-to-end security. This allows the application on the server to authenticate the client device. Without proper client authentication, the server application is at risk to a variety of MITM attacks. An MITM attack can be initiated by any attacker in the network.

7.4 Summary

Although the EAP-TLS authentic framework method offers the most secure settings, PEAP is much simpler to use. For the ODIN-W2 and NINA-W13/15 the total maximum size of the certificate is 8K (8192 bytes) – if the certificate is chained. Although certificates up to 4 kB (4096 bytes) can usually be handled, 2 kB (2048 bytes) certificates are recommended for normal use.

Both ODIN-W2 and NINA-W13/15 use the mbed TLS implementation <https://tls.mbed.org/>

In some application scenarios, it is possible to get debug firmware from the FAE with full TLS logging. The debug logging requires that the debug pin is available to capture it.

In some situations where a big, chained certificate is used it is often necessary to increase the in or outgoing buffer. This is necessary as mbed uses heap memory that is very limited. The default memory limits are low but can be increased to make in some situations. For example, to change the incoming TLS buffer to 10240, use the command `AT+UDCFG=101,10240`. To change the outgoing use from 102 to 3750, use the command `AT+UDCFG=102,3750`. These changes are only normally needed in special cases.

To change the min and max TLS versions, use the commands `UWCFG=24` and `UWCFG=25`. For example to set the min and max to TLS 1.2, use the command `UWCFG=24, 3` and `UWCFG=25, 3`.

- 802.1X is very complex and hard to debug
- No error codes are shown if the module is incorrectly configured
- Customers might not be able to share certificates that are used to reproduce issues
- Wireshark air traces and debug logs from u-blox module and AP are often required

8 Troubleshooting guide

8.1 Troubleshooting 802.1X connection issues

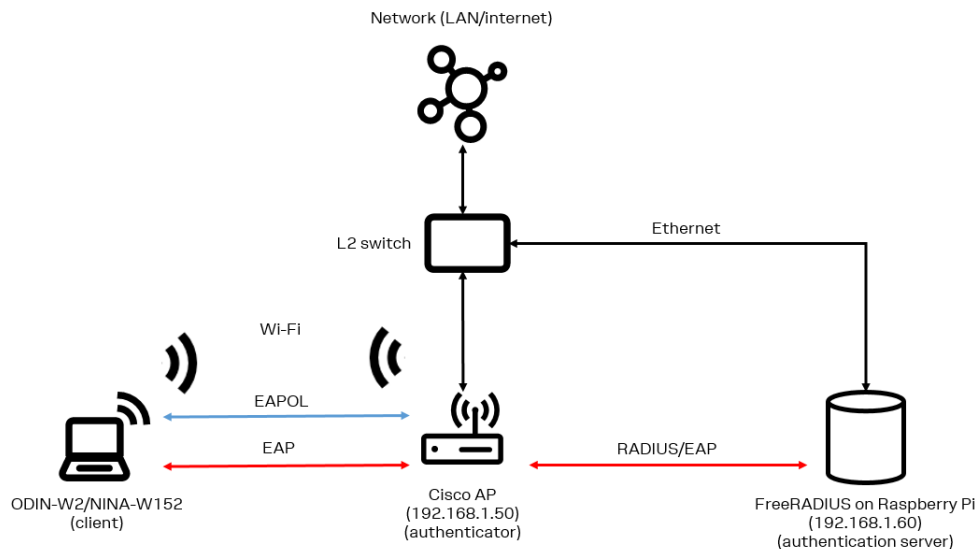


Figure 17: 802.1X test setup

8.1.1 Troubleshooting procedure

802.1X connection issues are complex to troubleshoot.

The following present example troubleshooting procedure for situation, where client fails to connect to the network using 802.1X.

1. Identify the issue and EAP method used.
2. Check URC's on short range module (client).
 - a) Does the authentication fail (+UUWLD: 0, 4)? Proceed to check RADIUS logs.
 - b) Is the network out-of-range (+UUWLD: 0, 2)? Check location of the client and AP configurations.

Note that the module needs to be in command, or extended data, mode to receive applicable URC's.

3. Check authentication logs on RADIUS authentication server.

If no entries on the logs, then the possible reason is communication issue with AP and server; AP has incorrect IP pointing the server, username and password are wrong, server has been incorrectly configured, EAP method has not been properly defined, and so on.

In most cases, the server logs reveal the reason for connection failure.

4. Check AP configuration and possible event logs.
5. Check client configuration.
 - a) Check that the certificates have been properly uploaded. Are they valid?
 - b) Check that the EAP has method been properly configured. Do the certificates and keys match the others on the server?
6. Capture and analyze traffic flow between the client, AP, and authentication server.

8.1.2 List of typical issues

Issue	Comment
Client is not able to authenticate and connect to the network.	Certificates are not valid or have not been properly installed on the Wi-Fi client and authentication server.
	Client has not been properly configured.
	Authentication server configuration is incorrect.
	Authentication server version may not support required ciphers or other features.
	Access point is missing configuration for authentication server, or the configuration is incorrect.
AP is not able to connect or access authentication server.	Authentication server configuration is incorrect.
Client is able to connect to network, but sometimes EAPOL between client and AP fails.	Possibly because the AP has not been patched for KRACK attacks. See https://www.krackattacks.com/ . This is particularly relevant for older AP types.

Table 7: List of typical issues for enterprise security

8.1.3 Logging available on RADIUS authentication server

The following logging capability is available on FreeRADIUS server. Similar logs can be found on other RADIUS server types also.

1. Run RADIUS on debug mode and observe login activity in real-time.

```
sudo freeradius -X
```

2. Open RADIUS log file.

```
sudo nano /var/log/freeradius/radius.log
```



Different logging capabilities need to be enabled on the authentication server.

8.1.3.1 Troubleshooting examples

Example #1

Issue: The module is unable to access the network. EAP-TLS is used for authentication.

Reason: AP has an incorrect password to access the authentication server.

Fix: Update the AP authentication server information on the AP.

Log analysis: Run the RADIUS server on debug mode. Observe the entries shown in the log.

```
Ready to process requests
(0) Received Access-Request Id 31 from 192.168.1.50:54255 to 192.168.1.60:1812 length 161
Dropping packet without response because of error: Received packet from 192.168.1.50 with
invalid Message-Authenticator! (Shared secret is incorrect.)
Waking up in 0.3 seconds.
(1) Received Access-Request Id 30 from 192.168.1.50:54255 to 192.168.1.60:1812 length 161
Dropping packet without response because of error: Received packet from 192.168.1.50 with
invalid Message-Authenticator! (Shared secret is incorrect.)
Waking up in 0.3 seconds.
(2) Received Access-Request Id 26 from 192.168.1.50:54255 to 192.168.1.60:1812 length 161
Dropping packet without response because of error: Received packet from 192.168.1.50 with
invalid Message-Authenticator! (Shared secret is incorrect.)
Waking up in 0.3 seconds.
(3) Received Access-Request Id 20 from 192.168.1.50:54255 to 192.168.1.60:1812 length 161
Dropping packet without response because of error: Received packet from 192.168.1.50 with
invalid Message-Authenticator! (Shared secret is incorrect.)
```


Example #2

Issue: The client is not able to access the network. EAP-TLS is used for authentication.

Reason: The client certificate is not valid.

Fix: Update the certificate on the client.

Log analysis: Run RADIUS server on debug mode. Observe the below presented entries on the log.

```
(6) eap: Calling submodule eap_tls to process data
(6) eap_tls: Continuing EAP-TLS
(6) eap_tls: Got final TLS record fragment (455 bytes)
(6) eap_tls: [eaptls verify] = ok
(6) eap_tls: Done initial handshake
(6) eap_tls: TLS_accept: SSLv3/TLS write server done
(6) eap_tls: <<< recv TLS 1.2 [length 034f]
(6) eap_tls: Creating attributes from certificate OIDs
(6) eap_tls:   TLS-Client-Cert-Serial := "2e401a756068f0e1cd8e581743207b560c3cc624"
(6) eap_tls:   TLS-Client-Cert-Expiration := "210530072118Z"
(6) eap_tls:   TLS-Client-Cert-Subject := "/C=FI/ST=Some-State/O=u-blox/OU=tester/CN=tester"
(6) eap_tls:   TLS-Client-Cert-Issuer := "/C=FI/ST=Some-State/L=Espoo/O=u-blox/OU=tester/CN=tester"
(6) eap_tls:   TLS-Client-Cert-Common-Name := "tester"
(6) eap_tls:   ERROR: SSL says error 20 : unable to get local issuer certificate
(6) eap_tls: >>> send TLS 1.2 [length 0002]
(6) eap_tls: ERROR: TLS Alert write:fatal:unknown CA
tls: TLS_accept: Error in error
(6) eap_tls: ERROR: Failed in __FUNCTION__ (SSL_read): error:1417C086:SSL
routines:tls_process_client_certificate:certificate verify failed
(6) eap_tls: ERROR: System call (I/O) error (-1)
(6) eap_tls: ERROR: TLS receive handshake failed during operation
(6) eap_tls: ERROR: [eaptls process] = fail
(6) eap: ERROR: Failed continuing EAP TLS (13) session. EAP sub-module failed
(6) eap: Sending EAP Failure (code 4) ID 6 length 4
(6) eap: Failed in EAP select
(6) [eap] = invalid
(6) } # authenticate = invalid
(6) Failed to authenticate the user
(6) Using Post-Auth-Type Reject
```

8.1.4 Wireshark capture analysis

For analysis, two traffic captures are needed: one showing the traffic between client and AP, and another one showing the traffic between AP and authentication server.

APs normally include a feature to capture traffic from different interfaces. For example, Wi-Fi radio.

Traffic to the authentication server can be captured on the line between the AP and server. For example, by connecting a PC to the L2 switch handling the traffic between Access point and server.

Figure 18 and Figure 19 show a successful EAP-TLS authentication.

No.	Time	Source	Destination	Protocol	Length	Info
1473	2020-04-15 12:50:45,270496653	192.168.1.50	192.168.1.60	RADIUS	211	Access-Request id=94
1474	2020-04-15 12:50:45,271004442	192.168.1.60	192.168.1.50	RADIUS	106	Access-Challenge id=94
1478	2020-04-15 12:50:45,387331787	192.168.1.50	192.168.1.60	RADIUS	343	Access-Request id=95
1479	2020-04-15 12:50:45,392623383	192.168.1.60	192.168.1.50	RADIUS	1110	Access-Challenge id=95
1480	2020-04-15 12:50:45,396307223	192.168.1.50	192.168.1.60	RADIUS	219	Access-Request id=96
1481	2020-04-15 12:50:45,396751532	192.168.1.60	192.168.1.50	RADIUS	1110	Access-Challenge id=96
1482	2020-04-15 12:50:45,400356854	192.168.1.50	192.168.1.60	RADIUS	219	Access-Request id=97
1483	2020-04-15 12:50:45,400750331	192.168.1.60	192.168.1.50	RADIUS	1110	Access-Challenge id=97
1484	2020-04-15 12:50:45,404318431	192.168.1.50	192.168.1.60	RADIUS	219	Access-Request id=98
1485	2020-04-15 12:50:45,404660797	192.168.1.60	192.168.1.50	RADIUS	574	Access-Challenge id=98
1557	2020-04-15 12:50:46,278922523	192.168.1.50	192.168.1.60	RADIUS	1255	Access-Request id=99
1558	2020-04-15 12:50:46,279256222	192.168.1.60	192.168.1.50	RADIUS	106	Access-Challenge id=99
1559	2020-04-15 12:50:46,282818434	192.168.1.50	192.168.1.60	RADIUS	1197	Access-Request id=100
1560	2020-04-15 12:50:46,300301869	192.168.1.60	192.168.1.50	RADIUS	161	Access-Challenge id=100
1561	2020-04-15 12:50:46,313533656	192.168.1.50	192.168.1.60	RADIUS	219	Access-Request id=101
1562	2020-04-15 12:50:46,314010187	192.168.1.60	192.168.1.50	RADIUS	215	Access-Accept id=101

> Frame 1562: 215 bytes on wire (1720 bits), 215 bytes captured (1720 bits) on interface eth0, id 0

> Ethernet II, Src: Raspberr_37:49:67 (dc:a6:32:37:49:67), Dst: Cisco_24:ed:70 (00:b1:e3:24:ed:70)

> Internet Protocol Version 4, Src: 192.168.1.60, Dst: 192.168.1.50

> User Datagram Protocol, Src Port: 1812, Dst Port: 60380

▼ RADIUS Protocol

Code: Access-Accept (2)

Packet identifier: 0x65 (101)

Length: 173

Authenticator: 94ac708d9aeecee7aa02122053badd31

[This is a response to a request in frame 1561]

[Time from request: 0.000476531 seconds]

▼ Attribute Value Pairs

> AVP: t=Vendor-Specific(26) l=58 vnd=Microsoft(311)

> AVP: t=Vendor-Specific(26) l=58 vnd=Microsoft(311)

> AVP: t=EAP-Message(79) l=6 Last Segment[1]

> AVP: t=Message-Authenticator(80) l=18 val=17589f94bc7d9666085f4cb617334e2f

> AVP: t=User-Name(1) l=13 val=my_username

Figure 18: Traffic between Access point and server on successful EAP-TLS authentication

No.	Time	Source	Destination	Protocol	Length	Info
471	2020-04-15 12:50:45,485499	u-blox_82:f7:36	Cisco_24:ed:72	802.11	59	Authentication, SN=1, FN=0, Flags=.....
472	2020-04-15 12:50:45,487061	Cisco_24:ed:72	u-blox_82:f7:36	802.11	56	Authentication, SN=0, FN=0, Flags=.....
474	2020-04-15 12:50:45,488089	u-blox_82:f7:36	Cisco_24:ed:72	802.11	145	Association Request, SN=2, FN=0, Flags=..... SSID=UBXHXfi_802_IX
477	2020-04-15 12:50:45,488089	Cisco_24:ed:72	u-blox_82:f7:36	802.11	165	Association Response, SN=0, FN=0, Flags=.....
487	2020-04-15 12:50:45,495160	Cisco_24:ed:72	u-blox_82:f7:36	802.11	48	Action, SN=0, FN=0, Flags=.....
488	2020-04-15 12:50:45,495253	Cisco_24:ed:72	u-blox_82:f7:36	EAP	63	Request, Identity
491	2020-04-15 12:50:45,496132	u-blox_82:f7:36	Cisco_24:ed:72	802.11	62	Action, SN=4, FN=0, Flags=.....
492	2020-04-15 12:50:45,496812	u-blox_82:f7:36	Cisco_24:ed:72	EAP	83	Response, Identity
498	2020-04-15 12:50:45,503053	Cisco_24:ed:72	u-blox_82:f7:36	EAP	59	Request, TLS EAP (EAP-TLS)
501	2020-04-15 12:50:45,518318	u-blox_82:f7:36	Cisco_24:ed:72	TLV1.2	197	Client Hello
502	2020-04-15 12:50:45,524675	Cisco_24:ed:72	u-blox_82:f7:36	EAP	1057	Request, TLS EAP (EAP-TLS)
504	2020-04-15 12:50:45,527347	u-blox_82:f7:36	Cisco_24:ed:72	EAP	73	Response, TLS EAP (EAP-TLS)
505	2020-04-15 12:50:45,528784	Cisco_24:ed:72	u-blox_82:f7:36	EAP	1057	Request, TLS EAP (EAP-TLS)
507	2020-04-15 12:50:45,531391	u-blox_82:f7:36	Cisco_24:ed:72	EAP	73	Response, TLS EAP (EAP-TLS)
508	2020-04-15 12:50:45,532789	Cisco_24:ed:72	u-blox_82:f7:36	EAP	1057	Request, TLS EAP (EAP-TLS)
510	2020-04-15 12:50:45,535383	u-blox_82:f7:36	Cisco_24:ed:72	EAP	73	Response, TLS EAP (EAP-TLS)
511	2020-04-15 12:50:45,536657	Cisco_24:ed:72	u-blox_82:f7:36	TLV1.2	525	Server Hello, Certificate, Certificate Request, Server Hello Done
530	2020-04-15 12:50:45,542576	u-blox_82:f7:36	Cisco_24:ed:72	802.11	53	Null function (No data), SN=5, FN=0, Flags=...P...T
537	2020-04-15 12:50:45,509059	u-blox_82:f7:36	Cisco_24:ed:72	EAP	1101	Response, TLS EAP (EAP-TLS)
538	2020-04-15 12:50:46,509045	u-blox_82:f7:36	Cisco_24:ed:72	802.11	53	Null function (No data), SN=6, FN=0, Flags=.....T
539	2020-04-15 12:50:46,511317	Cisco_24:ed:72	u-blox_82:f7:36	EAP	59	Request, TLS EAP (EAP-TLS)
541	2020-04-15 12:50:46,513841	u-blox_82:f7:36	Cisco_24:ed:72	TLV1.2	1045	Certificate, Client Key Exchange, Certificate Verify, Change Cipher Spec, Encrypted Handshake Message
543	2020-04-15 12:50:46,532387	Cisco_24:ed:72	u-blox_82:f7:36	TLV1.2	114	Change Cipher Spec, Encrypted Handshake Message
545	2020-04-15 12:50:46,544645	u-blox_82:f7:36	Cisco_24:ed:72	EAP	73	Response, TLS EAP (EAP-TLS)
546	2020-04-15 12:50:46,546241	Cisco_24:ed:72	u-blox_82:f7:36	EAP	57	Success
547	2020-04-15 12:50:46,546518	Cisco_24:ed:72	u-blox_82:f7:36	EAPOL	170	Key (Message 1 of 4)
550	2020-04-15 12:50:46,548783	u-blox_82:f7:36	Cisco_24:ed:72	EAPOL	184	Key (Message 2 of 4)
551	2020-04-15 12:50:46,549952	Cisco_24:ed:72	u-blox_82:f7:36	EAPOL	228	Key (Message 3 of 4)
553	2020-04-15 12:50:46,555492	u-blox_82:f7:36	Cisco_24:ed:72	EAPOL	162	Key (Message 4 of 4)
627	2020-04-15 12:50:47,449956	u-blox_82:f7:36	Cisco_24:ed:72	802.11	53	Null function (No data), SN=7, FN=0, Flags=...P...T
657	2020-04-15 12:50:48,137037	u-blox_82:f7:36	Cisco_24:ed:72	802.11	53	Null function (No data), SN=8, FN=0, Flags=...P...T
661	2020-04-15 12:50:48,341043	u-blox_82:f7:36	Cisco_24:ed:72	802.11	53	Null function (No data), SN=9, FN=0, Flags=...P...T
695	2020-04-15 12:50:49,344220	u-blox_82:f7:36	Cisco_24:ed:72	802.11	53	Null function (No data), SN=10, FN=0, Flags=...P...T
701	2020-04-15 12:50:49,550443	u-blox_82:f7:36	Cisco_24:ed:72	802.11	53	Null function (No data), SN=11, FN=0, Flags=...P...T
798	2020-04-15 12:50:54,453599	u-blox_82:f7:36	Cisco_24:ed:72	802.11	53	Null function (No data), SN=12, FN=0, Flags=...P...T
803	2020-04-15 12:50:54,668502	u-blox_82:f7:36	Cisco_24:ed:72	802.11	53	Null function (No data), SN=13, FN=0, Flags=...P...T
890	2020-04-15 12:50:58,385355	u-blox_82:f7:36	Cisco_24:ed:72	802.11	55	Disassociate, SN=14, FN=0, Flags=.....
891	2020-04-15 12:50:58,386059	u-blox_82:f7:36	Cisco_24:ed:72	802.11	53	Null function (No data), SN=15, FN=0, Flags=...P...T
892	2020-04-15 12:50:58,386170	Cisco_24:ed:72	u-blox_82:f7:36	802.11	41	Disassociate, SN=0, FN=0, Flags=.....

> Frame 546: 57 bytes on wire (456 bits), 57 bytes captured (456 bits)

> Radiotap Header v0, Length 15

> 802.11 radio information

> IEEE 802.11 QoS Data, Flags:F.

> Logical-Link Control

> 802.1X Authentication

▼ Extensible Authentication Protocol

Code: Success (3)

Id: 7

Length: 4

Figure 19: Traffic between client and Access point on successful EAP-TLS authentication

Figure 20 and Figure 21 show failed EAP-TLS authentication. Although it is not possible to identify this on these screen captures the failure is due to incorrect private key on the client.

((ip.src == 192.168.1.50) && (ip.dst == 192.168.1.60)) ((ip.src == 192.168.1.60) && (ip.dst == 192.168.1.50))						
No.	Time	Source	Destination	Protocol	Length	Info
623	2020-04-15 13:06:30,509440918	192.168.1.50	192.168.1.60	RADIUS	211	Access-Request id=102
624	2020-04-15 13:06:30,510029947	192.168.1.60	192.168.1.50	RADIUS	106	Access-Challenge id=102
640	2020-04-15 13:06:30,628277874	192.168.1.50	192.168.1.60	RADIUS	343	Access-Request id=103
641	2020-04-15 13:06:30,633462657	192.168.1.60	192.168.1.50	RADIUS	1110	Access-Challenge id=103
642	2020-04-15 13:06:30,637138719	192.168.1.50	192.168.1.60	RADIUS	219	Access-Request id=104
643	2020-04-15 13:06:30,637576658	192.168.1.60	192.168.1.50	RADIUS	1110	Access-Challenge id=104
644	2020-04-15 13:06:30,641143092	192.168.1.50	192.168.1.60	RADIUS	219	Access-Request id=105
645	2020-04-15 13:06:30,641522994	192.168.1.60	192.168.1.50	RADIUS	1110	Access-Challenge id=105
646	2020-04-15 13:06:30,645088391	192.168.1.50	192.168.1.60	RADIUS	219	Access-Request id=106
647	2020-04-15 13:06:30,645428479	192.168.1.60	192.168.1.50	RADIUS	574	Access-Challenge id=106
689	2020-04-15 13:06:31,523764257	192.168.1.50	192.168.1.60	RADIUS	1255	Access-Request id=107
690	2020-04-15 13:06:31,524330156	192.168.1.60	192.168.1.50	RADIUS	106	Access-Challenge id=107
691	2020-04-15 13:06:31,527877331	192.168.1.50	192.168.1.60	RADIUS	1197	Access-Request id=108
759	2020-04-15 13:06:32,572707097	192.168.1.60	192.168.1.50	RADIUS	86	Access-Reject id=108
1270	2020-04-15 13:06:42,672071887	192.168.1.50	192.168.1.60	RADIUS	211	Access-Request id=109
1271	2020-04-15 13:06:42,672421494	192.168.1.60	192.168.1.50	RADIUS	106	Access-Challenge id=109
1273	2020-04-15 13:06:42,790135798	192.168.1.50	192.168.1.60	RADIUS	343	Access-Request id=110
1274	2020-04-15 13:06:42,792283973	192.168.1.60	192.168.1.50	RADIUS	1110	Access-Challenge id=110
1275	2020-04-15 13:06:42,795941054	192.168.1.50	192.168.1.60	RADIUS	219	Access-Request id=111
1276	2020-04-15 13:06:42,796145051	192.168.1.60	192.168.1.50	RADIUS	1110	Access-Challenge id=111
1277	2020-04-15 13:06:42,799727596	192.168.1.50	192.168.1.60	RADIUS	219	Access-Request id=112
1278	2020-04-15 13:06:42,799903242	192.168.1.60	192.168.1.50	RADIUS	1110	Access-Challenge id=112
1279	2020-04-15 13:06:42,803458954	192.168.1.50	192.168.1.60	RADIUS	219	Access-Request id=113
1280	2020-04-15 13:06:42,803606934	192.168.1.60	192.168.1.50	RADIUS	574	Access-Challenge id=113
1346	2020-04-15 13:06:43,682466131	192.168.1.50	192.168.1.60	RADIUS	1255	Access-Request id=114
1347	2020-04-15 13:06:43,682779979	192.168.1.60	192.168.1.50	RADIUS	106	Access-Challenge id=114
1348	2020-04-15 13:06:43,688409126	192.168.1.50	192.168.1.60	RADIUS	1197	Access-Request id=115
1413	2020-04-15 13:06:44,706668099	192.168.1.60	192.168.1.50	RADIUS	86	Access-Reject id=115

> Frame 1413: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface eth0, id 0

> Ethernet II, Src: Raspberr_37:49:67 (dc:a6:32:37:49:67), Dst: Cisco_24:ed:70 (00:b1:e3:24:ed:70)

> Internet Protocol Version 4, Src: 192.168.1.60, Dst: 192.168.1.50

> User Datagram Protocol, Src Port: 1812, Dst Port: 60380

> RADIUS Protocol

Figure 20: Traffic between Access point and server on failed EAP-TLS authentication

((wlan.sa == d4:ca:5e:82:f7:36) && (wlan.da == 00:b1:e3:24:ed:72)) ((wlan.da == d4:ca:5e:82:f7:36) && (wlan.sa == 00:b1:e3:24:ed:72))						
No.	Time	Source	Destination	Protocol	Length	Info
719	2020-04-15 13:06:42,924529	Cisco_24:ed:72	u-blox_82:f7:36	802.11	48	Action, SN=0, FN=0, Flags=.....
720	2020-04-15 13:06:42,924630	Cisco_24:ed:72	u-blox_82:f7:36	EAP	63	Request, Identity
722	2020-04-15 13:06:42,924979	u-blox_82:f7:36	Cisco_24:ed:72	802.11	53	Null function (No data), SN=5, FN=0, Flags=.....T
725	2020-04-15 13:06:42,934271	u-blox_82:f7:36	Cisco_24:ed:72	EAP	83	Response, Identity
726	2020-04-15 13:06:42,934487	u-blox_82:f7:36	Cisco_24:ed:72	802.11	62	Action, SN=6, FN=0, Flags=.....
727	2020-04-15 13:06:42,935791	Cisco_24:ed:72	u-blox_82:f7:36	EAP	59	Request, TLS EAP (EAP-TLS)
735	2020-04-15 13:06:43,052511	u-blox_82:f7:36	Cisco_24:ed:72	TLV1.2	197	Client Hello
736	2020-04-15 13:06:43,055708	Cisco_24:ed:72	u-blox_82:f7:36	EAP	1057	Request, TLS EAP (EAP-TLS)
738	2020-04-15 13:06:43,058378	u-blox_82:f7:36	Cisco_24:ed:72	EAP	73	Response, TLS EAP (EAP-TLS)
739	2020-04-15 13:06:43,059556	Cisco_24:ed:72	u-blox_82:f7:36	EAP	1057	Request, TLS EAP (EAP-TLS)
741	2020-04-15 13:06:43,062168	u-blox_82:f7:36	Cisco_24:ed:72	EAP	73	Response, TLS EAP (EAP-TLS)
742	2020-04-15 13:06:43,063318	Cisco_24:ed:72	u-blox_82:f7:36	EAP	1057	Request, TLS EAP (EAP-TLS)
744	2020-04-15 13:06:43,065906	u-blox_82:f7:36	Cisco_24:ed:72	EAP	73	Response, TLS EAP (EAP-TLS)
745	2020-04-15 13:06:43,066905	Cisco_24:ed:72	u-blox_82:f7:36	TLV1.2	525	Server Hello, Certificate, Certificate Request, Server Hello Done
749	2020-04-15 13:06:43,272903	u-blox_82:f7:36	Cisco_24:ed:72	802.11	53	Null function (No data), SN=7, FN=0, Flags=...P...T
765	2020-04-15 13:06:43,944543	u-blox_82:f7:36	Cisco_24:ed:72	EAP	1101	Response, TLS EAP (EAP-TLS)
766	2020-04-15 13:06:43,944820	u-blox_82:f7:36	Cisco_24:ed:72	802.11	53	Null function (No data), SN=8, FN=0, Flags=.....T
767	2020-04-15 13:06:43,946184	Cisco_24:ed:72	u-blox_82:f7:36	EAP	59	Request, TLS EAP (EAP-TLS)
769	2020-04-15 13:06:43,950597	u-blox_82:f7:36	Cisco_24:ed:72	TLV1.2	1045	Certificate, Client Key Exchange, Certificate Verify, Change Cipher Spec, Encrypted Handshake Message
777	2020-04-15 13:06:44,154193	u-blox_82:f7:36	Cisco_24:ed:72	802.11	53	Null function (No data), SN=9, FN=0, Flags=...P...T
807	2020-04-15 13:06:45,177548	Cisco_24:ed:72	u-blox_82:f7:36	EAP	57	Failure
809	2020-04-15 13:06:45,177932	u-blox_82:f7:36	Cisco_24:ed:72	802.11	53	Null function (No data), SN=10, FN=0, Flags=.....T
813	2020-04-15 13:06:45,382839	u-blox_82:f7:36	Cisco_24:ed:72	802.11	53	Null function (No data), SN=11, FN=0, Flags=.....T
883	2020-04-15 13:06:48,216150	u-blox_82:f7:36	Cisco_24:ed:72	EAPOL	67	Start
884	2020-04-15 13:06:48,216464	u-blox_82:f7:36	Cisco_24:ed:72	802.11	53	Null function (No data), SN=12, FN=0, Flags=.....T
885	2020-04-15 13:06:48,221422	Cisco_24:ed:72	u-blox_82:f7:36	EAP	58	Request, Identity
887	2020-04-15 13:06:48,222497	u-blox_82:f7:36	Cisco_24:ed:72	EAP	83	Response, Identity
898	2020-04-15 13:06:48,427378	u-blox_82:f7:36	Cisco_24:ed:72	802.11	53	Null function (No data), SN=13, FN=0, Flags=...P...T

> Frame 807: 57 bytes on wire (456 bits), 57 bytes captured (456 bits)

> Radiotap Header v0, Length 15

> 802.11 radio information

> IEEE 802.11 QoS Data, Flags:F.

> Logical-Link Control

> 802.1X Authentication

> Extensible Authentication Protocol

Code: Failure (4)

Id: 6

Length: 4

Figure 21: Traffic between client and Access point on failed EAP-TLS authentication

8.1.5 Contacting u-blox support about connection issues

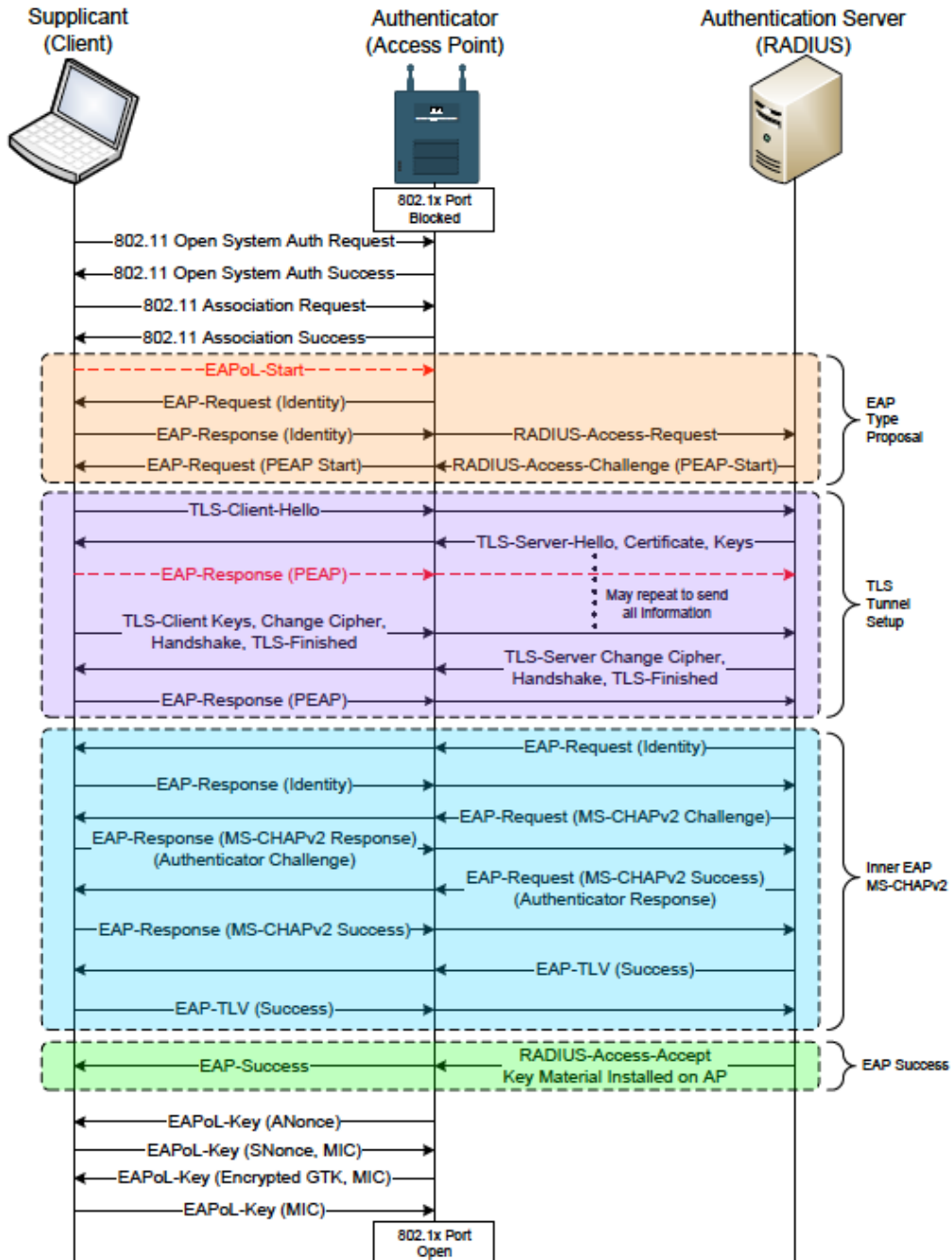
u-blox technical support need the following information to progress support issues related to 802.1X connection.

Although some information might not available or cannot be shared, please collect as much information about the problem as possible.

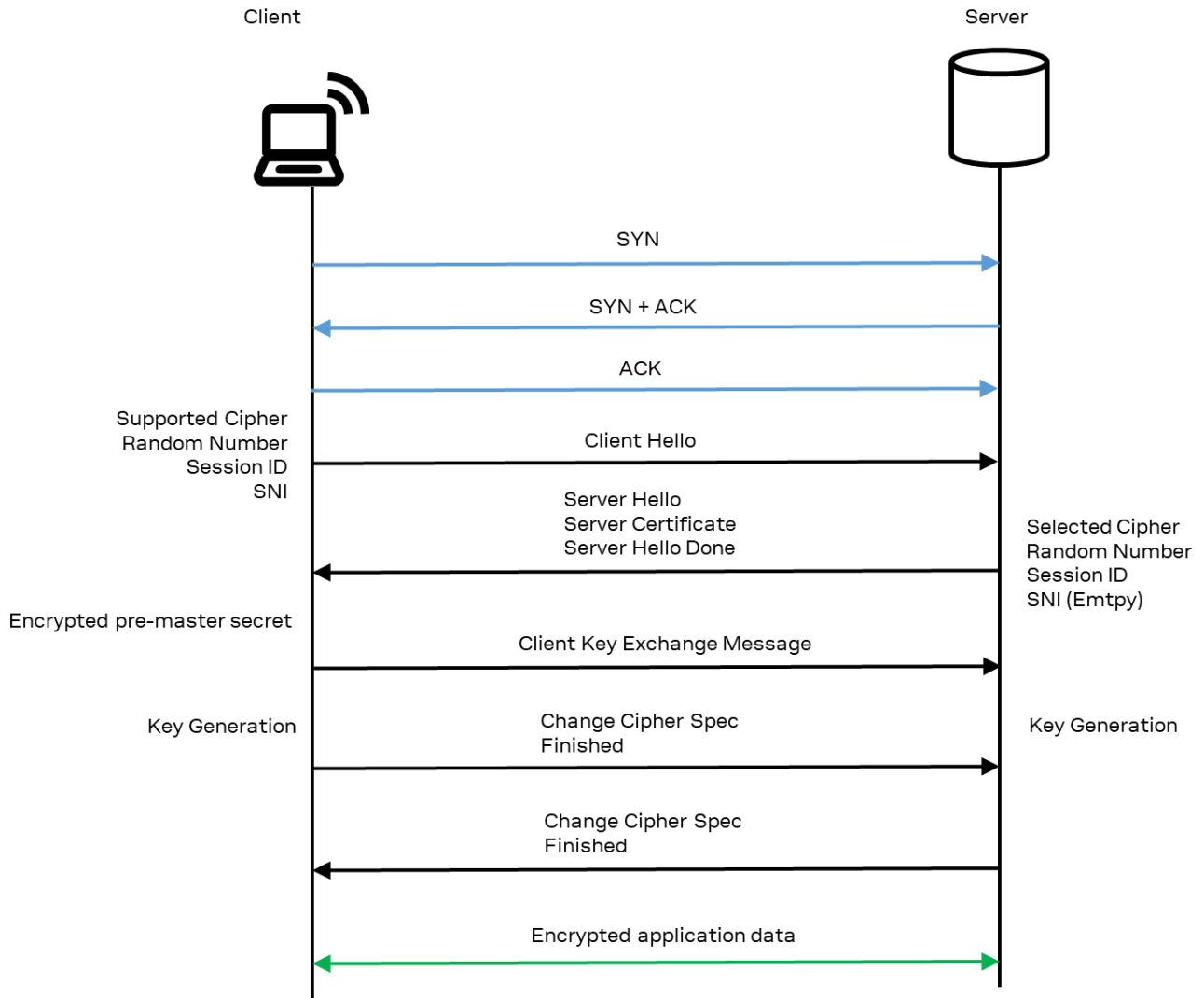
1. Detailed description of the issue:
 - a) Describe expected results versus actual results.
2. Setup description:
 - a) Detailed description of the test environment. Diagram showing all relevant devices and connection is preferred.
 - b) Wi-Fi AP type and manufacturer, configuration, and firmware version.
 - c) Wi-Fi client configuration and firmware version.
 - d) Security certificates and keys.
3. Log and trace files:
 - a) All available logs from the AP and authentication server.
 - b) Wireshark air traces showing traffic between the Wi-Fi client and AP.
 - c) Wireshark air traces showing the traffic between the AP and authentication server.

Appendix

A PEAP connection process



B TLS over TCP connection




C Glossary

Abbreviation	Definition
3DES	Triple Data Encryption Algorithm
AES	Advanced Encryption Standard
AP	Access Point
ASCII	American Standard Code for Information Interchange
CA	Certificate Authority
CBC	Cipher block chaining
CCM	Counter with CBC-MAC
CCMP	CCM mode Protocol
CPU	Central Processing Unit
CRL	Certificate revocation list
CSR	Certificate signing request
DC	Direct Current
DER	Distinguished Encoding Rules
DL	Down Link (Reception)
DTE	Data Terminal Equipment
EAP	Extensible Authentication Protocol
EAPOL	Extensible Authentication Protocol over LAN
ECDHE	Elliptic-curve Diffie–Hellman
GCM	Galois/Counter Mode
GTC	Generic Token Card
HMAC	Hash-based message authentication code
IETF	Internet Engineering Task Force
LEAP	Lightweight Extensible Authentication Protocol
MD5	Message-digest 5
MITM	Man-In-The-Middle attack
MSCHAP	Microsoft Challenge-Handshake Authentication Protocol
PEAP	Protected Extensible Authentication Protocol
PEM	Privacy-Enhanced Mail
PMK	Pairwise Master Key
PSK	Pre-shared key
RC4	Rivest Cipher 4
RSA	Rivest-Shamir-Adleman cryptosystem
SHA	Secure Hash Algorithms
SNI	Server Name Indication
SSL	Secure Socket Layer
TCP	Transport Control Protocol
TKIP	Temporal Key Integrity Protocol
TLS	Transport Layer Security
UTC	Coordinated Universal Time
WEP	Wired Equivalent Privacy
WPA	WiFi Protected Access
CBC-MAC	Cipher block chaining message authentication code

Table 8: Explanation of the abbreviations and terms used

Related documents

- [1] ODIN-W2 system integration manual, [UBX-14040040](#)
- [2] NINA-W1 system integration manual, [UBX-17005730](#)
- [3] u-connectXpress AT commands manual, [UBX-14044127](#)
- [4] u-blox package information guide, [UBX-14001652](#)
- [5] u-connectXpress user guide, [UBX-16024251](#)

 For product change notifications and regular updates of u-blox documentation, register on our website, www.u-blox.com.

Revision history

Revision	Date	Name	Comments
R01	7-Jul-2020	cmag	Initial release
R02	24-Sep-2020	flun	Added support for TLS 1.2 over TCP for NINA-W13/NINA-W15 using u-connectXpress 3.0. Editorial changes.
R03	29-Jan-2021	cmag	Added supported cipher suites and supported group suites for NINA-W13/NINA-W15 and ODIN-W2
R04	21-Jul-2021	cmag	Reworked introductory chapters with focus on describing the main features of WEP, WPA, WPA2, and WPA3 security protocols. Rationalized standards and topologies and EAP authentication methods. Highlighted depreciation of WEP 634/128 and WPA. Added more troubleshooting information.

Contact

For complete contact information, visit us at www.u-blox.com.

u-blox Offices

North, Central and South America

u-blox America, Inc.

Phone: +1 703 483 3180

E-mail: info_us@u-blox.com

Regional Office West Coast:

Phone: +1 408 573 3640

E-mail: info_us@u-blox.com

Technical Support:

Phone: +1 703 483 3185

E-mail: support@u-blox.com

Headquarters

Europe, Middle East, Africa

u-blox AG

Phone: +41 44 722 74 44

E-mail: info@u-blox.com

Support: support@u-blox.com

Asia, Australia, Pacific

u-blox Singapore Pte. Ltd.

Phone: +65 6734 3811

E-mail: info_ap@u-blox.com

Support: support_ap@u-blox.com

Regional Office Australia:

Phone: +61 3 9566 7255

E-mail: info_au@u-blox.com

Support: support_ap@u-blox.com

Regional Office China (Beijing):

Phone: +86 10 68 133 545

E-mail: info_cn@u-blox.com

Support: support_cn@u-blox.com

Regional Office China (Chongqing):

Phone: +86 23 6815 1588

E-mail: info_cn@u-blox.com

Support: support_cn@u-blox.com

Regional Office China (Shanghai):

Phone: +86 21 6090 4832

E-mail: info_cn@u-blox.com

Support: support_cn@u-blox.com

Regional Office China (Shenzhen):

Phone: +86 755 8627 1083

E-mail: info_cn@u-blox.com

Support: support_cn@u-blox.com

Regional Office India:

Phone: +91 80 405 092 00

E-mail: info_in@u-blox.com

Support: support_in@u-blox.com

Regional Office Japan (Osaka):

Phone: +81 6 6941 3660

E-mail: info_jp@u-blox.com

Support: support_jp@u-blox.com

Regional Office Japan (Tokyo):

Phone: +81 3 5775 3850

E-mail: info_jp@u-blox.com

Support: support_jp@u-blox.com

Regional Office Korea:

Phone: +82 2 542 0861

E-mail: info_kr@u-blox.com

Support: support_kr@u-blox.com

Regional Office Taiwan:

Phone: +886 2 2657 1090

E-mail: info_tw@u-blox.com

Support: support_tw@u-blox.com