Secure Socket Layer (SSL) for Micro-Controller over Wireless LAN

This document illustrates how to secure network link by using SSL. The example setups a SSL connection with Apache Web server, and transmit/receive data securely via crypto processing.

Table of Contents

1 Introduction

This document illustrates how to secure network link by using SSL and HTTP. PolarSSL 1.3.8 is used to support SSL connection. The SSL client sample code is provided to setup a SSL connection with Apache Web server, and transmit/receive data securely via crypto processing. The SSL server sample code is provided to setup a SSL connection with SSL client, and transmit/receive data securely via crypto processing. As the web server sample code, a connection must be established between the browser and the web server before accessing AP configuration page. HTTP download is useful when implementing OTA to download firmware from remote site. Following sections explain how to execute the SSL and HTTP sample codes and all the provided PolarSSL configuration files which can be used with these SSL and HTTP examples.

2 SSL Client Configuration

A SSL client sample is already implemented in ssl_client.c to demonstrate the SSL connection. The SSL client can be executed by interactive mode command. To support this SSL client AT command, the definitions of CONFIG_SSL_CLIENT in platform_opts.h must be modified as following.

```
/* platform_opts.h */
#define CONFIG_SSL_CLIENT                    1
```

If need to verify the certificate of the server or the client, the ssl_client_ext.c must be included, so the definition of SSL_CLIENT_EXT in ssl_client.c must be modified as following.

```
/* ssl_client.c */
#define SSL_CLIENT_EXT                       1
```

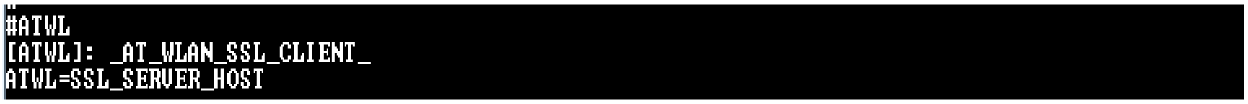
If need to verify the server certificate, the definition of SSL_VERIFY_SERVER in ssl_client_ext.c must be modified as following.

```
/* ssl_client_ext.c */
#define SSL_VERIFY_SERVER                    1
```

If need to verify the client certificate, the definition of SSL_VERIFY_CLIENT in ssl_client_ext.c must be modified as following.

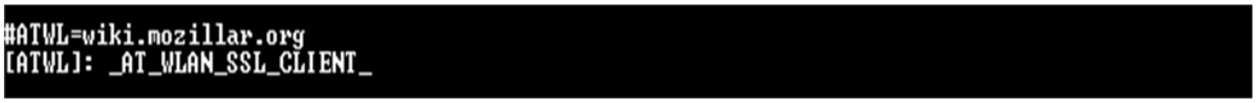
```
/* ssl_client_ext.c */
#define SSL_VERIFY_CLIENT                    1
```

By specifying an IP address in ATWL command, the SSL client will start to connect the SSL server related this address via HTTPS on port 443. The following is the information of ATWL command.



```
#ATWL
[ATWL]: _AT_WLAN_SSL_CLIENT_
ATWL=SSL_SERVER_HOST
```

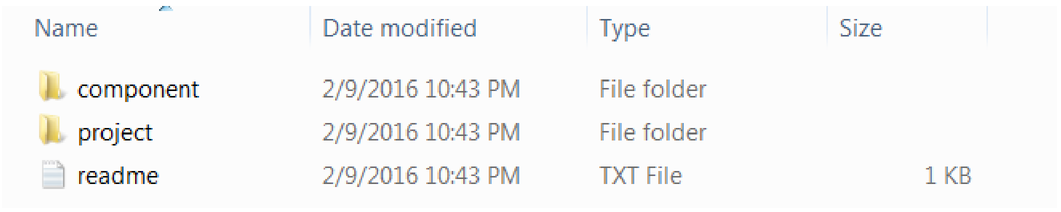
A quicker start to evaluate SSL connection is to use the exiting AT command. The following is the information of example ATWL command.



```
#ATWL=wiki.mozillar.org
[ATWL]: _AT_WLAN_SSL_CLIENT_
```

3 SSL Server Configuration

A SSL server sample is already implemented in the patch of SSL server example to demonstrate the SSL connection. Copy and replace the patch files under sdk-ameba1-v3.4b* to standard SDK sdk-ameba1-v3.4b\, the content of patch package is shown as following.



Name	Date modified	Type	Size
component	2/9/2016 10:43 PM	File folder	
project	2/9/2016 10:43 PM	File folder	
readme	2/9/2016 10:43 PM	TXT File	1 KB

Modify SERVER_PORT and response content in example_ssl_server.c based on your SSL server.

```
/* example_ssl_server.c */
#define SERVER_PORT                         443
```

To support this SSL server example, the definitions of POLARSSL_CERTS_C and POLARSSL_SSL_SRV_C in config_rsa.h must be modified as following.

```
/* config_rsa. h */
#define POLARSSL_CERTS_C
#define POLARSSL_SSL_SRV_C
```

The definition of CONFIG_EXAMPLE_SSL_SERVER in platform_opts.h must be modified as following.

```
/* platform_opts. h */
#define CONFIG_EXAMPLE_SSL_SERVER        1
```

To reduce the ssl_handshake time, you can move the bignum.c from SDRAM to SRAM.

4 HTTP Server Configuration

A HTTP server sample is already implemented in webserver.c located in component\common\utilities to demonstrate the http connection. The details of web server configuration refer to UM0014.

4.1 Configuration of SoftAP Mode

The following ATCMDs are used to configure SoftAP mode. The details of ATCMD refer to AN0025.

```
# ATW3 = ssid
# ATW4 = password
# ATW5 = channel
# ATWA
```

4.2 Configuration of Web Server

Firstly, set CONFIG_WEBSERVER to 1 to enable web server in **platform_opts.h**. Secondly, enter ATCMD “**ATWE**” to start webserver.

If CONFIG_READ_FLASH defined in atcmd_wifi.c is set to 1, AP settings from “ATWA” will be saved to FLASH DATA_SECTOR. And if CONFIG_READ_FLASH defined in webserver.c is set to 1, AP settings from webserver will be saved to Flash DATA_SECTOR. After power off/on or reset, enter command “ATWE”, SoftAP will be started with these settings.

```
/* platform_opts. h */
#define  CONFIG_WEBSERVER                1

/* webserver.c */
#define CONFIG_READ_FLASH                1
/* atcmd_wifi.c */
#define CONFIG_READ_FLASH                1
```

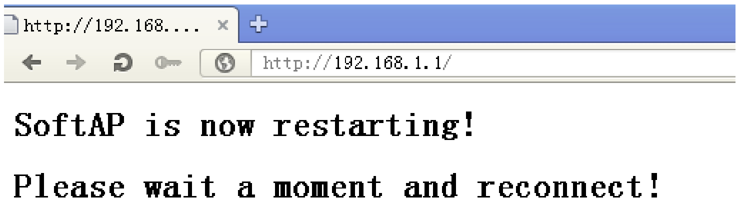
ATCMD to start webserver.

```
# ATWE
```

When the Wi-Fi device of the client PC has already connected to the AP, please open browser and enter http://192.168.1.1/.

There are two security types for the web server, open and WPA2. SSID length should be between 1 to 31 characters and Password length between 8 to 31 characters.

After submitting, page will jump to the Wait Page, and the device AP is going to restart with new settings. The Wait Page is shown below. The AP will restart after about 3~5 seconds.



SoftAP is now restarting!

Please wait a moment and reconnect!

5 HTTP Download Configuration

A HTTP download sample is already implemented in http_download.c located in component\common\example to demonstrate the http download connection.

It is useful when implementing OTA to download firmware from remote site. Replace socket read/write with ssl_read/ssl_write can achieve https download.

```
/* platform_opts. h */
#define  CONFIG_EXAMPLE_HTTP_DOWNLOAD                    1
```

Modify the SERVER_HOST, SERVER_PORT and RESOURCE content in example_http_download.c based on your SSL server.

```
/* example_http_download.c */
#define SERVER_HOST             "192.168.1.50"
#define SERVER_PORT              80
#define RESOURCE                "/repository/IOT/Project_Cloud_A.bin"
```

6 PolarSSL Configuration

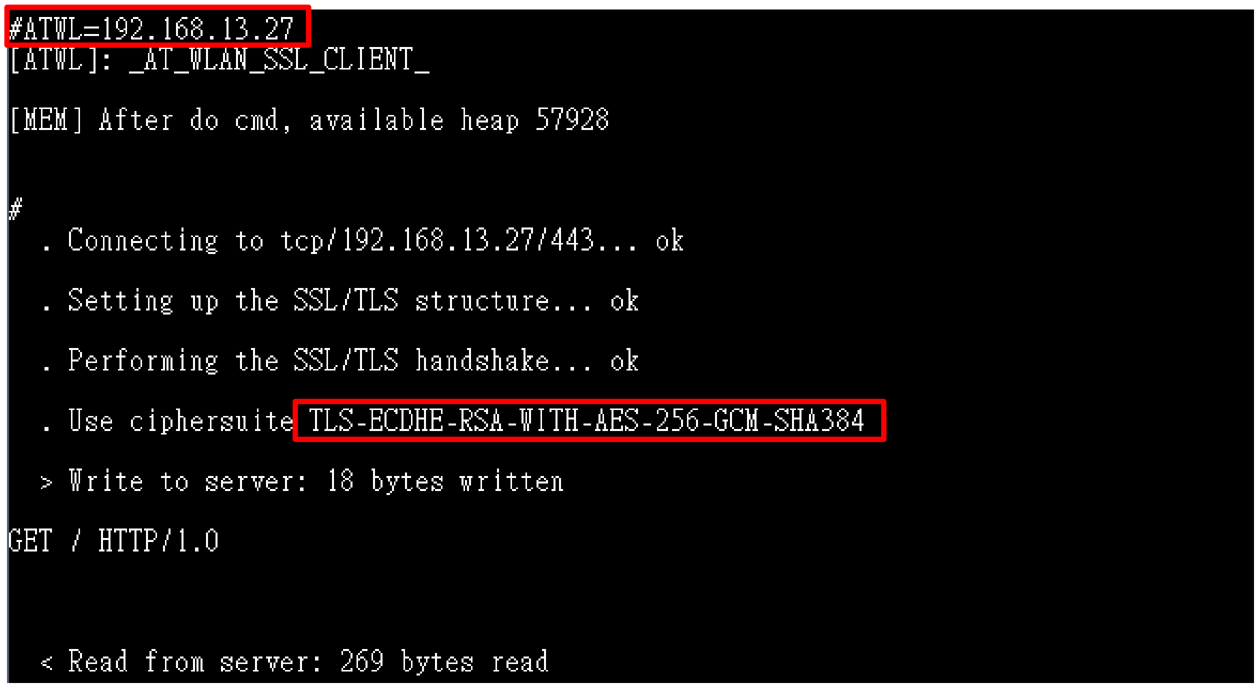
Some configuration files are provided to configure PolarSSL library to support different cipher suites. The following sub-sections explain the PolarSSL configurations for all cipher suites and RSA-AES-SHA cipher suites.

6.1 Configuration for All Cipher Suites

PolarSSL library provides several cipher suites for SSL connection. The provided configuration file of config_all.h enables all the supported cipher suites of PolarSSL 1.3.8. To enable config_all.h configuration for PolarSSL library, the definitions in polarssl/config.h should be modified as following.

```
/* polarssl/config.h */
#define CONFIG_SSL_RSA      0
```

The following is the executing of SSL client when configuring PolarSSL to use config_all.h. In this example, SSL server in 192.168.13.27 is connected by ATWL command. SSL server determines to use TLS-ECDHE-RSA-WITH-AES-256-GCM-SHA384 from all the proposed cipher suites during SSL handshake. SSL client requests to get homepage of web server via SSL connection successfully.

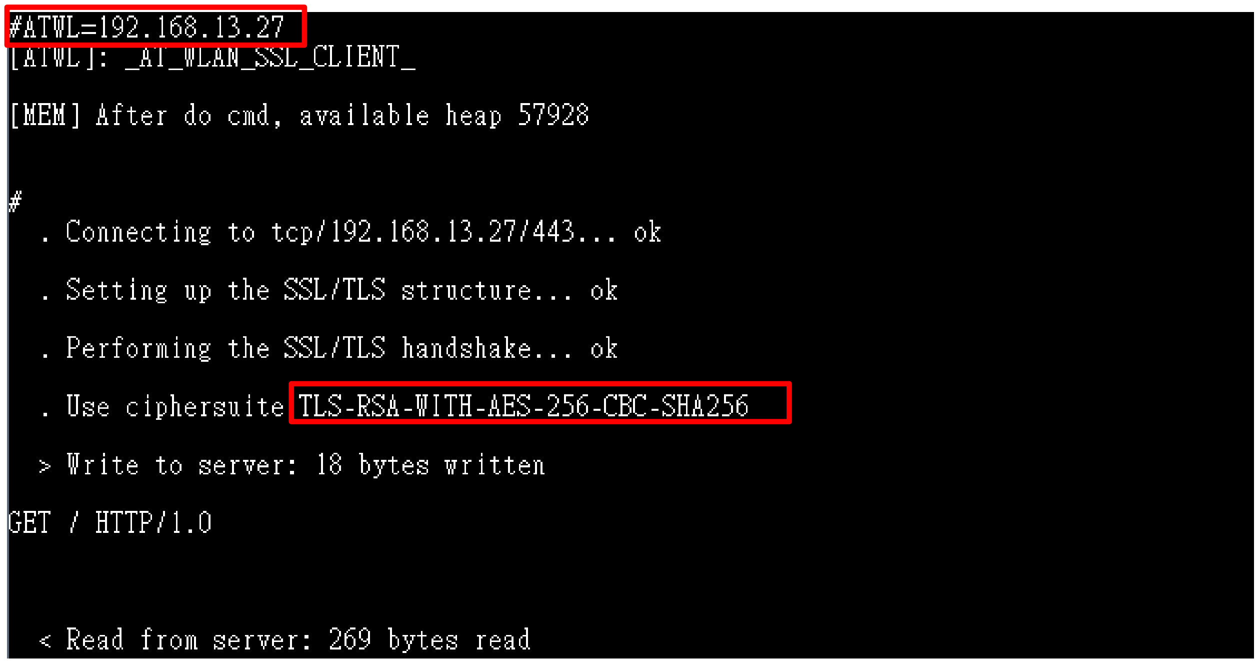

```
#ATWL=192.168.13.27
[ATWL]: _AT_WLAN_SSL_CLIENT_

[MEM] After do cmd, available heap 57928


#
  . Connecting to tcp/192.168.13.27/443... ok

  . Setting up the SSL/TLS structure... ok

  . Performing the SSL/TLS handshake... ok

  . Use ciphersuite TLS-ECDHE-RSA-WITH-AES-256-GCM-SHA384

  > Write to server: 18 bytes written

GET / HTTP/1.0


  < Read from server: 269 bytes read
```

6.2 Configuration for RSA-AES-SHA Cipher Suite

To reduce the memory requirement when using PolarSSL library, a configuration file of config_rsa.h is provided to only enable all the cipher suites of RSA-AES-SHA. Therefore, only cipher suites using RSA-AES-SHA will be proposed in SSL handshake if the config_rsa.h configuration is adopted. To enable config_rsa.h configuration for PolarSSL library, the definitions in polarssl/config.h should be modified as following.

```
/* polarssl/config.h */
#define CONFIG_SSL_RSA      1
```

The following is the executing of SSL client when configuring PolarSSL to use config_rsa.h. In this example, SSL server in 192.168.13.27 is connected by ATWL command. SSL server determines to use TLS-RSA-WITH-AES-256-CBC-SHA256 from all the proposed RSA-AES-SHA cipher suites during SSL handshake. SSL client requests to get homepage of web server via SSL connection successfully.



```
#ATWL=192.168.13.27
[ATWL]: _AT_WLAN_SSL_CLIENT_

[MEM] After do cmd, available heap 57928


#
  . Connecting to tcp/192.168.13.27/443... ok

  . Setting up the SSL/TLS structure... ok

  . Performing the SSL/TLS handshake... ok

  . Use ciphersuite TLS-RSA-WITH-AES-256-CBC-SHA256

  > Write to server: 18 bytes written

GET / HTTP/1.0


  < Read from server: 269 bytes read
```

7 Memory Configuration

The config_all.h using default PolarSSL library configuration will require large heap memory for SSL input/output buffer (16384 bytes). To reduce the memory usage, customized configuration is enabled in config_rsa.h. The definitions of SSL_MAX_CONTENT_LEN for SSL input/output buffer is modified according to the requirement of RSA cipher suites as following. This value may need to be increased based on the cipher suite determined by server or the size of data transferred from server.

```
/* polarssl/config_rsa.h */
#define SSL_MAX_CONTENT_LEN           4096
```

Beside of heap usage, task stack size should also be considered when using PolarSSL library. For example, the definition of POLARSSL_DEBUG_C enabled in config_all.h and config_rsa.h will enable debug functions of PolarSSL library, but it will also require more task stack size. It will increase about 1k bytes of stack size for config_rsa.h configuration compared with that when disabling POLARSSL_DEBUG_C. Therefore, STACKSIZE of SSL client task could be modified based on the PolarSSL configuration.

```
/* ssl_client.c */
#define STACKSIZE    1150

/* config_all.h, config_rsa.h */
#define POLARSSL_DEBUG_C
```