ESP8266 SSL 加密使用手册

Version 1.1

Espressif Systems IOT Team

Table of Contents

1. 前言

本文主要介绍基于 ESP8266 物联网模块的 SDK SSL 加密使用方法，将分别介绍 ESP8266 作为 SSL server 和 ESP8266 作为 SSL client 的使用方法。

SSL 功能需要占用大量内存，请开发者在上层应用程序确保内存足够。在将 SSL 缓存设置为 8KB 的情况下，SSL 功能至少需要 22KB 的空间，由于服务器的证书大小不同，所需空间可能更大。

更多 ESP8266 的信息，请访问：http://bbs.espressif.com/

2. ESP8266 作为 SSL server

ESP8266 作为 SSL server 时，提供加密证书的制作脚本，生成 SSL 加密所需的头文件。用户可以参考 IOT_Demo 中 `#define SERVER_SSL_ENABLE` 宏定义的代码，实现 SSL server 功能。

CA 认证功能默认关闭，用户可调用接口 `espconn_secure_ca_enable` 使能 CA 认证。

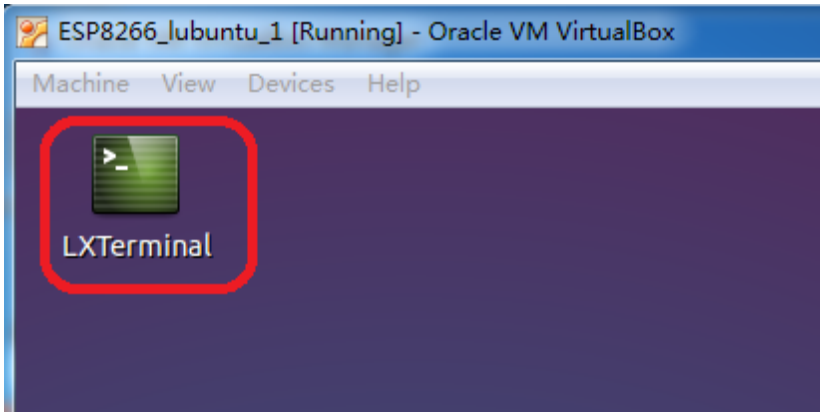
2.1. 证书制作

(1) 用户可向 Espressif Systems 申请 ESP8266 SSL server 证书制作脚本 "**makefile.sh**"，将脚本拷贝到 lubuntu 虚拟机共享路径下。 lubuntu 虚拟机编译环境可在 Espressif BBS 下载，

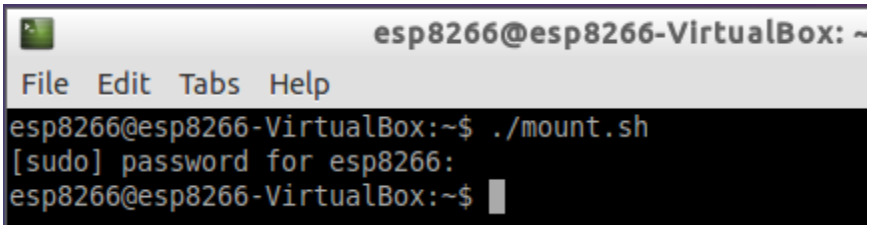
- 下载链接： http://bbs.espressif.com/viewtopic.php?f=21&t=86

(2) 挂载共享路径。

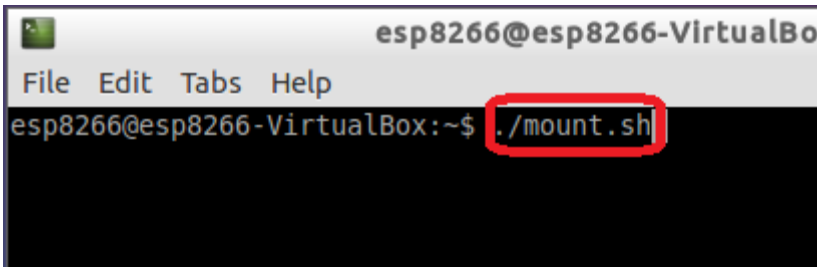
- 打开虚拟机桌面的 "LXTerminal"



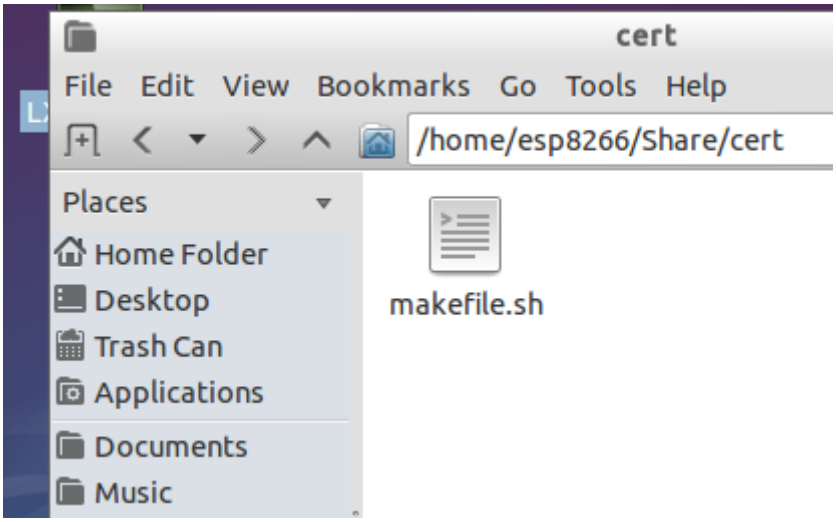
- 输入指令 `./mount.sh` ，回车



- 输入密码 espressif ，回车



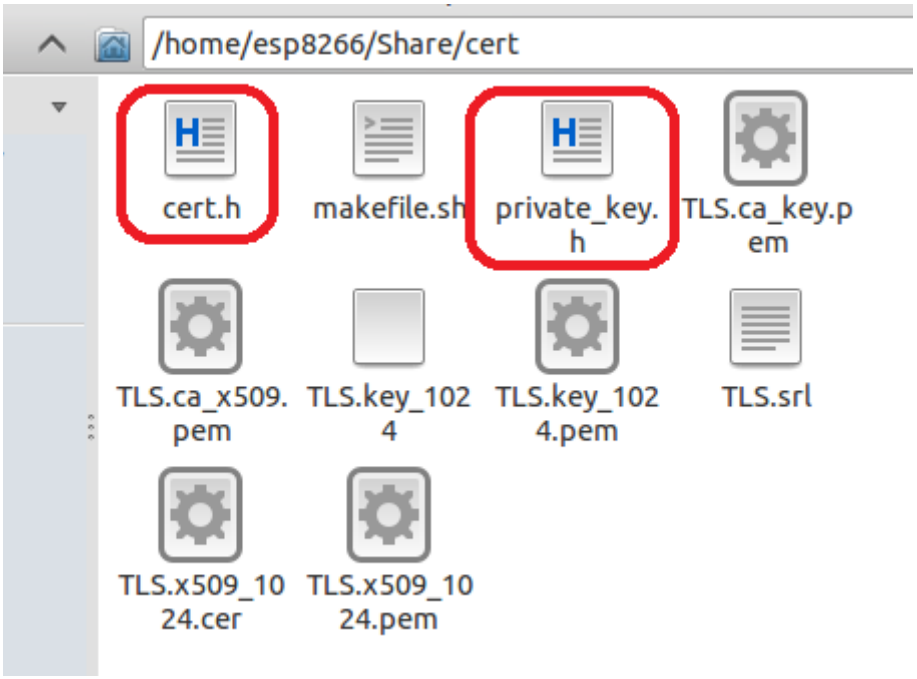
(3) 在虚拟机打开共享文件夹，看到证书制作脚本，挂载成功。



(4) 在脚本所在路径，运行脚本，指令 `./makefile.sh` ，生成 `cert.h` 和 `private_key.h`，参考 IOT_Demo 使用这两个头文件即可。

```
esp8266@esp8266-VirtualBox:~$ cd /home/esp8266/Share/cert
esp8266@esp8266-VirtualBox:~/Share/cert$ ./makefile.sh
```

运行成功，生成如下：



注意事项

- 证书制作脚本 makefile.sh 文件中的 IP 地址，请改为用户实际的 server IP ，如下图：

```
cat > certs.conf << EOF
[ req ]
distinguished_name     = req_distinguished_name
prompt                 = no

[ req_distinguished_name ]
 O                      = $PROJECT_NAME
 CN                     = 127.0.0.1
EOF
```

- 证书制作脚本 makefile.sh 默认为 1024 的加密算法位数长度；如需使用 512 的加密算法位数，请修改 makefile.sh 文件中的 1024 改为 512：

```
# private key generation
openssl genrsa -out TLS.ca_key.pem 1024
openssl genrsa -out TLS.key_1024.pem 1024

# convert private keys into DER format
openssl rsa -in TLS.key_1024.pem -out TLS.key_1024 -outform DER

# cert requests
openssl req -out TLS.ca_x509.req -key TLS.ca_key.pem -new \
            -config ./ca_cert.conf
openssl req -out TLS.x509_1024.req -key TLS.key_1024.pem -new \
            -config ./certs.conf

# generate the actual certs.
openssl x509 -req -in TLS.ca_x509.req -out TLS.ca_x509.pem \
            -sha1 -days 5000 -signkey TLS.ca_key.pem
openssl x509 -req -in TLS.x509_1024.req -out TLS.x509_1024.pem \
            -sha1 -CAcreateserial -days 5000 \
            -CA TLS.ca_x509.pem -CAkey TLS.ca_key.pem

# some cleanup
rm TLS*.req
rm *.conf

openssl x509 -in TLS.ca_x509.pem -outform DER -out TLS.ca_x509.cer
openssl x509 -in TLS.x509_1024.pem -outform DER -out TLS.x509_1024.cer

#
# Generate the certificates and keys for encrypt.
#
```

- 证书制作脚本 makefile.sh 生成的默认 SSL server 证书由 Espressif Systems 颁发，并非由 CA 颁发。如果用户需要 CA 认证，有以下两种方式：
  - 用户将脚本 makefile.sh 运行生成的 TLS . ca_x509.cer 导入 SSL client，并参考后文 **3.1 证书制作** 使用脚本 "**make_cert.py**" 将 CA 文件生成 esp_ca_cert.bin 烧写到 flash 对应的地址；
  - 用户使用签发的证书，参考脚本 makefile.sh，自行生成对应的 cert.h 和 private_key.h，并参考后文 **3.1 证书制作** 使用脚本 "**make_cert.py**" 将 CA 文件生成 "esp_ca_cert.bin" 烧写到 flash 对应的地址。

3. ESP8266 作为 SSL client

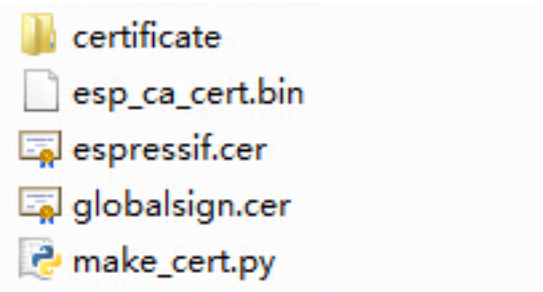
用户可以参考 IOT_Demo 中 `#define CLIENT_SSL_ENABLE` 宏定义的代码，实现 SSL client 功能。ESP8266 作为 SSL client 时，可支持 CA 认证。

CA 认证功能默认关闭，用户可调用接口 `espconn_secure_ca_enable` 使能 CA 认证。

3.1. 证书制作

(1) 用户可向 Espressif Systems 申请 ESP8266 SSL client 证书合成脚本 "**make_cert.py**"，将合成脚本与 CA 文件放在同一目录下。

(2) 运行脚本 "make_cert.py" 将合成同一目录下的 CA 文件（目前支持最多 2 个 CA 证书）生成 esp_ca_cert.bin， esp_ca_cert.bin 的烧录位置由接口 `espconn_secure_ca_enable` 设置，用户可自行定义。



3.2. CA 认证流程

(1) ESP8266 作为 client 发起 TCP 连接，成功建立，读取 flash 中的 esp_ca_cert.bin (CA 证书)，加载到对应的 SSL ctx，目前限定的 CA 最大支持数目为 2 个。

(2) ESP8266 发起 TLS handshake，获取 SSL server 签发的证书，与 ESP8266 支持的 CA 进行验证：

- 若失败，则断开当前连接；
- 若成功，完成验证。

4. 软件接口

SSL 系列软件接口与普通 TCP 软件接口，在 SDK 底层是两套不同的处理流程，因此，请不要混用两种软件接口。SSL 连接时，仅支持使用：

- espconn_secure_XXX 系列接口；
- espconn_regist_XXX 系列注册回调的接口；
- espconn_port 获得一个空闲端口。

本文仅介绍 espconn_secure_XXX 系列接口，更多的软件接口介绍，请参考 ESP8266 编程手册 "2C-ESP8266__SDK__Programming Guide"

SSL 连接，用户可参考 BBS 提供的 Demo http://bbs.espressif.com/viewtopic.php?f=21&t=389

4.1. espconn_secure_ca_disable

功能：

关闭 SSL CA 认证功能

注意：

- CA 认证功能，默认关闭
- 如需调用本接口，请在加密（SSL）连接建立前调用：

  在 espconn_secure_accept (ESP8266 作为 TCP SSL server) 之前调用；

  或者 espconn_secure_connect (ESP8266 作为 TCP SSL client) 之前调用

函数定义：

```
bool espconn_secure_ca_disable (uint8 level)
```

参数：

uint8 level : 设置 ESP8266 SSL server/client：

0x01 SSL client；

0x02 SSL server；

0x03 SSL client 和 SSL server

返回：

true : 成功

false : 失败

4.2. espconn_secure_ca_enable

功能：

开启 SSL CA 认证功能

注意：

- CA 认证功能，默认关闭
- 如需调用本接口，请在加密（SSL）连接建立前调用：

  在 espconn_secure_accept （ESP8266 作为 TCP SSL server）之前调用；

  或者 espconn_secure_connect （ESP8266 作为 TCP SSL client）之前调用

函数定义：

```
bool espconn_secure_ca_enable (uint8 level, uint16 flash_sector)
```

参数：

uint8 level ：设置 ESP8266 SSL server/client：

0x01 SSL client；

0x02 SSL server；

0x03 SSL client 和 SSL server

uint16 flash_sector ：设置 CA 证书（esp_ca_cert.bin）烧录到 flash 的位置，例如，参数传入 0x3B，则对应烧录到 flash 0x3B000

返回：

true ：成功

false ：失败

4.3. espconn_secure_accept

功能：

创建 SSL TCP server，侦听 SSL 握手。

注意：

- 本接口只能调用一次
- 仅支持建立一个 SSL server，并且仅支持连入一个 SSL client。
- 如果 SSL 加密一包数据大于 espconn_secure_set_size 设置的缓存空间，ESP8266 无法处理，SSL 连接断开，进入 espconn_reconnect_callback

函数定义：

```
sint8 espconn_secure_accept(struct espconn *espconn)
```

参数：

```
struct espconn *espconn : 对应网络连接的结构体
```

返回：

```
0        : 成功
Non-0    : 失败，返回错误码
           ESPCONN_ARG – 未找到参数 espconn 对应的 TCP 连接
           ESPCONN_MEM – 空间不足
           ESPCONN_ISCONN – 连接已经建立
```

4.4. espconn_secure_set_size

功能：

设置加密（SSL）数据缓存空间的大小

注意：

- 默认缓存大小为 2KBytes；如需更改，请在加密（SSL）连接建立前调用：
  - 在 espconn_secure_accept（ESP8266 作为 TCP SSL server）之前调用；
  - 或 espconn_secure_connect（ESP8266 作为 TCP SSL client）之前调用

函数定义：

```
bool espconn_secure_set_size (uint8 level, uint16 size)
```

参数：

```
uint8 level : 设置 ESP8266 SSL server/client：
              0x01  SSL client；
              0x02  SSL server；
              0x03  SSL client 和 SSL server
uint16 size : 加密数据缓存的空间大小，取值范围：1 ~ 8192，单位：字节
              默认值为 2048
```

返回：

```
true   : 成功
false  : 失败
```

4.5. espconn_secure_get_size

功能：

查询加密（SSL）数据缓存空间的大小

函数定义：

```
sint16 espconn_secure_get_size (uint8 level)
```

参数：

`uint8 level` ：设置 ESP8266 SSL server/client：

- `0x01` SSL client；
- `0x02` SSL server；
- `0x03` SSL client 和 SSL server

返回：

加密（SSL）数据缓存空间的大小

4.6. espconn_secure_connect

功能：

加密（SSL）连接到 TCP SSL server（ESP8266 作为 TCP SSL client）

注意：

- 目前 ESP8266 作为 SSL client 仅支持一个连接，本接口如需多次调用，请先调用 `espconn_secure_disconnect` 断开前一次连接，再建立下一个 SSL 连接；
- 如果 SSL 加密一包数据大于 `espconn_secure_set_size` 设置的缓存空间，ESP8266 无法处理，SSL 连接断开，进入 `espconn_reconnect_callback`

函数定义：

```
sint8 espconn_secure_connect (struct espconn *espconn)
```

参数：

`struct espconn *espconn` ：对应网络连接的结构体

返回：

`0` ：成功

`Non-0` ：失败，返回错误码

- `ESPCONN_ARG` – 未找到参数 `espconn` 对应的 TCP 连接
- `ESPCONN_MEM` – 空间不足
- `ESPCONN_ISCONN` – 传输已经建立

4.7. espconn_secure_send

功能：

发送加密数据（SSL）

注意：

请在上一包数据发送完成，进入 espconn_sent_callback 后，再发下一包数据。

函数定义：

```
sint8 espconn_secure_send (
        struct espconn *espconn,
        uint8 *psent,
        uint16 length
)
```

参数：

struct espconn *espconn ： 对应网络连接的结构体

uint8 *psent ： 发送的数据

uint16 length ： 发送的数据长度

返回：

0 ： 成功

Non-0 ： 失败，返回错误码 ESPCONN_ARG – 未找到参数 espconn 对应的 TCP 连接

4.8. espconn_secure_disconnect

功能：

断开加密 TCP 连接(SSL)

函数定义：

```
sint8 espconn_secure_disconnect(struct espconn *espconn)
```

参数：

struct espconn *espconn ： 对应网络连接的结构体

返回：

0 ： 成功

Non-0 ： 失败，返回错误码 ESPCONN_ARG – 未找到参数 espconn 对应的 TCP 连接