XR806 Mbed TLS 网络模块

开发指南

版本号：1.0

发布时间：2020-11-10

版本历史

版本	日期	责任人	版本描述
1.0	2020-11-10	AWA 1096	创建文档。

目录

表格目录

1 前言

1.1 文档简介

本文档介绍了 XR806 平台上 Mbed TLS 模块的使用方法。

1.2 目标读者

XR806 开发及维护人员

1.3 适用范围

此文档适用于 XR806 SDK，支持 XR806 系列芯片产品

1.4 文档约定

1.4.1 标志说明

本文档采用各种醒目的标志来表示在操作过程中应该特别注意的地方，这些标志的含义如下：

标识	说明
警告	该标志后的说明应给予格外关注，如果不遵守，可能会导致人员受伤或死亡。
注意	提醒操作中应注意的事项。不当的操作可能会损坏器件，影响可靠性、降低性能等。
说明	为准确理解文中指令、正确实施操作而提供的补充或强调信息。
窍门	一些容易忽视的小功能、技巧。了解这些功能或技巧能帮助解决特定问题或者节省操作时间。

2 概述

2.1 背景说明

Mbed TLS 库支持一组加密组件，包括 SSL/TLS 通信模块、TCP/IP 通信模块、哈希模块、RNG 模块、Cipher 模块、PK 模块、X.509 模块，并且这些模块可以通过配置文件进行独立使用。SSL/TLS 模块提供整体安全框架；TCP/IP 模块提供下层网络通信接口；Hash 模块提供散列算法；RNG 模块提供随机数；Cipher 模块提供对称加密算法；X.509 提供证书相关接口。

XR806 SDK 移植了 Mbed TLS 2.16.0 版本，为了用户开发方便、快速搭建安全服务。移植过程中，在 Mbed TLS 接口库的基础上对证书加密通信接口做了进一步的封装，本文对封装后的接口做了详细的介绍。如果接口不满足读者需求，读者可以直接调用 Mbed TLS 提供的原始接口，源代码中也提供了大量的例程供参考。

2.2 规格特性

Mbed TLS 模块提供了以下功能。

表 2-1 Mbed TLS 模块的规格特性

规格类型	支持规格	规格描述	备注
版本支持	2.16.0	Mbed TLS 2.16.0	
功能	Server	服务器功能，接收/发送数据	
	Client	客户端功能，发送/接收数据	
加解密套件	TLS-RSA-WITH-AES-256-CBC-SHA256	加解密套件之一	
	TLS-RSA-WITH-AES-256-CBC-SHA	加解密套件之一	
	TLS-RSA-WITH-AES-128-CBC-SHA256	加解密套件之一	
	MBEDTLS_TLS_RSA_WITH_AES_256_CBC_SHA	加解密套件之一	
	其它		未测试验证

2.3 文件位置

以 SDK 包为根目录，本网络协议涉及到的主要文件位置如下。

表 2-2 Mbed TLS 网络协议的文件位置

组件名	文件分类	文件位置
Mbed TLS	源码文件	sdk/src/net/mbedtls-2.2.0 sdk/src/net/mbedtls-2.16.0

组件名	文件分类	文件位置
	头文件	sdk/include/net/mbedtls
	示例工程	sdk/project/common/cmd/tls

说明

XR806 SDK 可在以下 GitHub 仓库获取：https://github.com/XradioTech/xr806_sdk.git

3 应用说明

3.1 应用简述

Mbed TLS 网络协议已经内嵌到 XR806 SDK 中，通过配置以及函数接口调用即可使用。

3.2 配置说明

表 3-1 XR806 Mbed TLS 模块配置列表

配置项	配置说明
配置文件选择	设置说明： 此项配置选择使用的配置文件 设置位置： /src/net/mbedtls-2.16.0/Makefile 设置方式： 修改宏 MBEDTLS_CONFIG_FILE 的值，如： CC_FLAGS += -DMBEDTLS_CONFIG_FILE='<config-xr-mini-cliserv.h>'
调试打印接口	设置说明： 此项配置使能/关闭调试信息 设置位置： 配置文件（配置文件的位置，由配置项“配置文件选择”决定） 设置方式： 定义宏 MBEDTLS_DEBUG_C，如： #define MBEDTLS_DEBUG_C
输入输出 buffer 大小	设置说明： 此项配置输入输出的 buffer 大小 设置位置： 配置文件（配置文件的位置，由配置项“配置文件选择”决定） 设置方式： 修改宏 MBEDTLS_SSL_MAX_CONTENT_LEN 的值，如： #define MBEDTLS_SSL_MAX_CONTENT_LEN (16*1024)

3.3 接口说明

表 3-2 Mbed TLS 模块接口简介

接口名	简要介绍
mbedtls_socket 接口	本接口用于创建一个 TLS socket，属于外部接口。
mbedtls_closesocket 接口	本接口用于关闭一个 TLS socket，属于外部接口。
mbedtls_init_context 接口	本接口用于初始化上下文，属于外部接口
mbedtls_deinit_context 接口	本接口用于反初始化上下文，属于外部接口
mbedtls_config_context 接口	本接口用于配置信息，属于外部接口
mbedtls_handshake 接口	本接口用于执行握手动作，属于外部接口
Mbed TLS 数据发送接口	本接口用于发送数据，属于外部接口
Mbed TLS 数据接收接口	本接口用于接收数据，属于外部接口

3.3.1 mbedtls_socket 接口

3.3.1.1 mbedtls_socket

表 3-3 mbedtls_socket 接口函数说明

信息项	说明
原型	mbedtls_sock* mbedtls_socket(int nonblock);
功能	创建 Mbed TLS 会话句柄
参数	int nonblock 含义解释：阻塞标志位。 使用说明：0：阻塞；1：非阻塞。
返回值	mbedtls_sock 类型指针

3.3.2 mbedtls_closesocket 接口

3.3.2.1 mbedtls_closesocket

表 3-4 mbedtls_closesocket 接口函数说明

信息项	说明
原型	int mbedtls_closesocket(mbedtls_sock* fd);
功能	关闭 Mbed TLS 连接
参数	mbedtls_sock* fd 含义解释：Mbed TLS 会话句柄。 使用说明：mbedtls_socket 创建的会话句柄
返回值	0：成功 其它：失败

3.3.3 mbedtls_init_context 接口

3.3.3.1 mbedtls_init_context

表 3-5 mbedtls_init_context 接口函数说明

信息项	说明
原型	mbedtls_context* mbedtls_init_context(int client);
功能	初始化 Mbed TLS 上下文
参数	int client 含义解释：当前为客户端还是服务端。 使用说明：1：客户端；0：服务端
返回值	mbedtls_context 类型指针

3.3.4 mbedtls_deinit_context 接口

3.3.4.1 mbedtls_deinit_context

表 3-6 mbedtls_deinit_context 接口函数说明

信息项	说明
原型	void mbedtls_deinit_context(mbedtls_context *context);
功能	反初始化 Mbed TLS 上下文
参数	mbedtls_context *context 含义解释：Mbed TLS 上下文指针。 使用说明：mbedtls_init_context 创建的上下文指针
返回值	无

3.3.5 mbedtls_config_context 接口

3.3.5.1 mbedtls_config_context

表 3-7 mbedtls_config_context 接口函数说明

信息项	说明
原型	int mbedtls_config_context(mbedtls_context *context, void *param, int verify);
功能	配置 Mbed TLS 上下文，将客户端/服务器配置好的证书结构体配置到 Mbed TLS
参数	mbedtls_context *context 含义解释：Mbed TLS 上下文指针。 使用说明：mbedtls_init_context 创建的上下文指针 void *param 含义解释：指向证书结构体（server/client）的指针。 使用说明：证书结构体的成员说明，请参阅表 3-8 security_server 结构体的成员说明和表 3-9 security_client 结构体的成员说明
返回值	0：成功

信息项	说明
	其它：失败

服务器模式下，需要三个文件：CA 证书，自己的证书，自己的私钥。CA 证书和自己的证书构成一个证书链。结构体为 security_server，成员说明如下：

表 3-8 security_server 结构体的成员说明

成员项	说明
char *pCa;	含义解释：CA 证书 使用说明：赋值即可
unsigned int nCa;	含义解释：CA 证书数据长度 使用说明：赋值即可
char *pCert;	含义解释：自己的证书 使用说明：赋值即可
unsigned int nCert;	含义解释：自己证书数据长度 使用说明：赋值即可
char *pKey;	含义解释：自己的私钥 使用说明：赋值即可
unsigned int nKey;	含义解释：自己私钥的长度 使用说明：赋值即可

客户端模式下，需要三个文件：CA 证书，自己的证书，自己的私钥。CA 证书用来验证服务器，自己的证书和私钥用于双向验证。下面结构体中，pCa 变量和 certs 中的 pCa 变量是一样的，都是 CA 证书，这样重复设计的原因是因为在绝大部分连接中，都是单向验证，不需要设置客户端证书（只有在双向验证中才需要配置 certs 变量），这样设计便于代码编写方便。结构体为 security_client，成员说明如下：

表 3-9 security_client 结构体的成员说明

成员项	说明
char *pCa;	含义解释：CA 证书 使用说明：赋值即可
unsigned int nCa;	含义解释：CA 证书数据长度 使用说明：赋值即可

成员项	说明
security_server certs;	含义解释：服务器证书结构体 使用说明：参阅表 3-11 security_server 结构体的成员说明

3.3.6 mbedtls_handshake 接口

3.3.6.1 mbedtls_handshake

表 3-10 mbedtls_handshake 接口函数说明

信息项	说明
原型	int mbedtls_handshake(mbedtls_context *context, mbedtls_sock* fd);
功能	执行握手流程
参数	mbedtls_context *context 含义解释：Mbed TLS 上下文指针。 使用说明：mbedtls_init_context 创建的上下文指针 mbedtls_sock* fd 含义解释：Mbed TLS 会话句柄。 使用说明：mbedtls_socket 创建的会话句柄
返回值	0：成功 其它：失败

3.3.7 Mbed TLS 数据发送接口

3.3.7.1 mbedtls_send

表 3-11 mbedtls_send 接口函数说明

信息项	说明
原型	int mbedtls_send(mbedtls_context *context, char *buf, int len);
功能	发送数据
参数	mbedtls_context *context 含义解释：Mbed TLS 上下文指针。 使用说明：mbedtls_init_context 创建的上下文指针 char *buf 含义解释：数据 buffer。 使用说明：赋值即可 int len 含义解释：数据的长度。 使用说明：赋值即可
返回值	0：成功

信息项	说明
	其它：失败

3.3.8 Mbed TLS 数据接收接口

3.3.8.1 mbedtls_recv

表 3-12 mbedtls_recv 接口函数说明

信息项	说明
原型	int mbedtls_recv(mbedtls_context *context, char *buf, int len);
功能	接收数据
参数	mbedtls_context *context 含义解释：Mbed TLS 上下文指针。 使用说明：mbedtls_init_context 创建的上下文指针 char *buf 含义解释：数据 buffer。 使用说明：赋值即可 int len 含义解释：数据的长度。 使用说明：赋值即可
返回值	0：成功 其它：失败

3.3.8.2 mbedtls_recv_pending

表 3-13 mbedtls_recv_pending 接口函数说明

信息项	说明
原型	int mbedtls_recv_pending(mbedtls_context *context);
功能	查询 Mbed TLS 层剩下未读取的数据长度
参数	mbedtls_context *context 含义解释：Mbed TLS 上下文指针。 使用说明：mbedtls_init_context 创建的上下文指针
返回值	剩余数据的长度

4 示例说明

Mbed TLS 接口用于进行一次 Mbed TLS 加密连接，下面以 Mbed TLS 连接为例，简要说明 Mbed TLS 接口的使用。

4.1 示例简介

Mbed TLS 示例的演示的目的是简要介绍 Mbed TLS 接口的基本使用方法，演示的功能为访问服务器下载文件。

4.1.1 获取方式

Mbed TLS 示例只有串口命令代码，位于 XR806 SDK 的/project/common/cmd/tls 目录。

说明

XR806 SDK 可在以下 GitHub 仓库获取：https://github.com/XradioTech/xr806_sdk.git

4.1.2 准备工作

Mbed TLS 示例工程的硬件准备如下。

1. 评估板：运行示例工程代码。
2. 串口线：连接评估板的 Uart0 插针，用于 console 控制台的输入输出。
3. PC 机：用于镜像烧录和 console 控制的输入输出。

Mbed TLS 的软件准备，包括烧写工具、代码编译和烧写操作，请参见《XR806_SDK_快速入门指南》

4.1.3 操作步骤

首先需要在一台设备上搭建 TLS 服务器。开发板完成烧写，复位，通过串口命令连接网络后，通过 Mbed TLS 的命令进行连接。

表 4-1 Mbed TLS 示例工程的命令列表

命令	说明
快速配置网络	命令格式： net sta config <ssid> [psk] 命令解释： 快速配置被连接 AP 的 SSID 和密码 参数说明： ssid：被连接 AP 的 ssid psk：连连接 AP 的密码，如果为开放网络，则不需要此项。
启用网络	命令格式：

命令	说明
	net sta enable 命令解释： 开始连接 AP，需要先完成网络配置
连接服务器	命令格式： net tls -p <port> -c -n <host> -t <time> 命令解释： 连接服务器 参数说明： port：端口 host：服务器的 ip 地址。 time：测试时间，单位 ms

4.2 效果展示

在评估板中运行 HTTPC 示例程序后，在控制台中会打印出连接的各个阶段，并显示数据接收过程，如下所示。

```
$ <net> <tls> <test>
Seeding the random number generator.
ok
Loading the CA root certificate .
 ok (0 skipped)
Connecting to 192.168.51.102 (10000).
 ok
Setting up the SSL/TLS structure...
 ok
Performing the SSL/TLS handshake...
 ok(TLS-RSA-WITH-AES-256-CBC-SHA256)
Verifying peer X.509 certificate...
 failed
  !
Write to server.
 86 bytes written
Read from server:
 86 bytes read
Read data success
<net> <tls> <response : success>
```

4.3 实现流程

Mbed TLS 支持创建服务器和客户端，来发送、接收数据。本节会介绍创建服务器和客户端的代码实现流程。

4.3.1 客户端实现流程

第一步：获取网络上下文

```
mbedtls_sock*netfd = mbedtls_socket(0);
```

第二步：创建并初始化 TLS 上下文

```
mbedtls_context *pContext = (mbedtls_context *)mbedtls_init_context(0);
```

第三步： 配置 TLS 上下文

```
if ((ret = mbedtls_config_context(pContext, (void *) &client_param,
MBEDTLS_SSL_CLIENT_VERIFY_LEVEL)) != 0) {
    HC_ERR(("https: config failed.."));
    return -1;
}
```

第四步： 建立连接

```
if ((ret = mbedtls_connect(pContext, (mbedtls_sock*) &net_fd, ServerAddress, namelen, hostname)) != 0)
{
    HC_ERR(("https: connect failed.."));
    return -1;
}
```

第五步：协商握手

```
if ((ret = mbedtls_handshake(g_pContext, &g_httpc_net_fd)) != 0)
    return -1;
```

第六步：发送/接收数据

```
if ((ret = mbedtls_send(g_pContext, buf, len)) < 0)
    return -1;

if ((ret = mbedtls_recv(g_pContext, buf, len)) < 0)
    return -1;
```

第七步：释放 tls 上下文

```
mbedtls_deinit_context(g_pContext);
```

第八步：释放网络上下文

```
mbedtls_closesocket(net_fd);
```

4.3.2 服务器实现流程

第一步：获取网络上下文

```
mbedtls_sock remote_fd;
mbedtls_sock*local_fd = mbedtls_socket(0);
```

第二步：创建并初始化 TLS 上下文

```
mbedtls_context *pContext = (mbedtls_context *)mbedtls_init_context(1);
```

第三步： 配置 TLS 上下文

```
mbedtls_config_context(ssl_context,    (void *) &server_param, MBEDTLS_SSL_SERVER_VERIFY_LEVEL);
```

第四步： 建立连接

```
ret = mbedtls_accept(mbedtls_context *context, mbedtls_sock *local_fd, &remote_fd);
```

第五步：协商握手

```
if ((ret = mbedtls_handshake(g_pContext, &g_httpc_net_fd)) != 0)
    return -1;
```

第六步：发送/接收数据

```
if ((ret = mbedtls_send(g_pContext, buf, len)) < 0)
    return -1;

if ((ret = mbedtls_recv(g_pContext, buf, len)) < 0)
    return -1;
```

第七步：释放 tls 上下文

```
mbedtls_deinit_context(g_pContext);
```

第八步：释放网络上下文

```
mbedtls_closesocket(local_fd);
mbedtls_closesocket(remote_fd);
```

著作权声明

商标声明

免责声明