XR871 MBEDTLS Developer Guide

Revision 1.0

Sep 11, 2017

Declaration

THIS DOCUMENTATION IS THE ORIGINAL WORK AND COPYRIGHTED PROPERTY OF XRADIO TECHNOLOGY ("XRADIO"). REPRODUCTION IN WHOLE OR IN PART MUST OBTAIN THE WRITTEN APPROVAL OF XRADIO AND GIVE CLEAR ACKNOWLEDGEMENT TO THE COPYRIGHT OWNER.

THE INFORMATION FURNISHED BY XRADIO IS BELIEVED TO BE ACCURATE AND RELIABLE. XRADIO RESERVES THE RIGHT TO MAKE CHANGES IN CIRCUIT DESIGN AND/OR SPECIFICATIONS AT ANY TIME WITHOUT NOTICE. XRADIO DOES NOT ASSUME ANY RESPONSIBILITY AND LIABILITY FOR ITS USE. NOR FOR ANY INFRINGEMENTS OF PATENTS OR OTHER RIGHTS OF THE THIRD PARTIES WHICH MAY RESULT FROM ITS USE. NO LICENSE IS GRANTED BY IMPLICATION OR OTHERWISE UNDER ANY PATENT OR PATENT RIGHTS OF XRADIO. THIS DATASHEET NEITHER STATES NOR IMPLIES WARRANTY OF ANY KIND, INCLUDING FITNESS FOR ANY PARTICULAR APPLICATION.

THIRD PARTY LICENCES MAY BE REQUIRED TO IMPLEMENT THE SOLUTION/PRODUCT. CUSTOMERS SHALL BE SOLELY RESPONSIBLE TO OBTAIN ALL APPROPRIATELY REQUIRED THIRD PARTY LICENCES. XRADIO SHALL NOT BE LIABLE FOR ANY LICENCE FEE OR ROYALTY DUE IN RESPECT OF ANY REQUIRED THIRD PARTY LICENCE. XRADIO SHALL HAVE NO WARRANTY, INDEMNITY OR OTHER OBLIGATIONS WITH RESPECT TO MATTERS COVERED UNDER ANY REQUIRED THIRD PARTY LICENCE.

Revision History

Version	Data	Summary of Changes
1.0	2017-09-11	Initial Version

Table 1-1 Revision History

Contents

Tables

Figures

1 模块概要

1.1 功能介绍

Mbedtls 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。

XR871 移植的版本为 mbedtls2.2.0，为了用户开发方便、快速搭建安全服务。移植过程中，在 mbedtls 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 mbedtls 提供的原始接口，源代码中也提供了大量的例程供参考。

1.2 代码位置

module	file	location
MBEDTLS	source	sdk/src/net/mbedtls-2.2.0
	header	sdk/include/net/mbedtls

表 1-1 mbedtls 文件结构

2 模块接口

module	function	detail
MBEDTLS	mbedtls_socket	mbedtls_sock* mbedtls_socket(int nonblock) Purpose： 创建网络上下文。 Param： nonblock: 非阻塞标志 Returns： 返回上下文指针
	mbedtls_init_context	mbedtls_context* mbedtls_init_context(int client) Purpose： 创建 tls 上下文。 Param： client: 当前服务为客户端或服务端。 Returns： 返回 tls 上下文
	mbedtls_deinit_context	void mbedtls_deinit_context(mbedtls_context *context) Purpose： 释放 tls 上下文。 Param： context: tls 上下文指针 Returns： 无
	mbedtls_closesocket	int mbedtls_closesocket(mbedtls_sock* fd) Purpose： 释放网络上下文。 Param： fd: 网络上下文指针 Returns： 返回状态 (0: ok)
	mbedtls_config_context	int mbedtls_config_context(mbedtls_context *context, void *param, int verify) Purpose： 配置 tls 上下文。 Param： context: tls 上下文指针 param: 指向安全信息（server/client）的指针 verify: 证书校验模式 Returns： 返回状态 (0: ok)
	mbedtls_handshake	int mbedtls_handshake(mbedtls_context *context, mbedtls_sock* fd) Purpose： 执行 tls 握手流程。 Param： context: tls 上下文指针 fd: 网络上下文指针 Returns： 返回状态 (0: ok)
	mbedtls_send	int mbedtls_send(mbedtls_context *context, char *buf, int len)

		Purpose： 发送数据。 Param： context: tls 上下文指针 buf: 发送 buffer 的指针 len: 发送数据的长度 Returns： 返回状态 (0: ok)
	mbedtls_recv	int mbedtls_recv(mbedtls_context *context, char *buf, int len) Purpose： 接收数据。 Param： context: tls 上下文指针 buf: 接收 buffer 的指针 len: 接收 buffer 的长度 Returns： 返回接收的字节长度
	mbedtls_recv_pending	int mbedtls_recv_pending(mbedtls_context *context) Purpose： 用于查询 tls 层剩下未读取的数据。 Param： context: tls 上下文指针 Returns： 返回字节长度
	mbedtls_connect	int mbedtls_connect(mbedtls_context *context, mbedtls_sock* fd, struct sockaddr *name, int namelen, char *hostname) Purpose： 创建连接。 Param： context: tls 上下文指针 fd: 指定的网络上下文 name: 服务器端的 sockaddr namelen: name 的长度 hostname: server 的域名 Returns： 返回状态 (0: ok)
	mbedtls_accept	int mbedtls_accept(mbedtls_context *context, mbedtls_sock *local_fd, mbedtls_sock *remote_fd) Purpose： 接受连接。 Param： context: tls 上下文指针 local_fd: 本地网络上下文 len: 远端网络上下文 Returns： 返回连接的状态

3 模块接口例程

本节提供上节介绍接口的示例，描述接口的使用方法及流程，文中代码皆为参考代码，不能直接运行。shttpd 与 httpclient 模块的安全传输机制，都是基于上述接口的实现，读者可以参考 HTTPMbedTLSWrapper.c、io_ssl.c 的调用过程。

注意：本节不会描述 mbedtls 原生接口的使用，有兴趣的读者可以参考官方实例（sdk/src/net/mbedtls-2.2.0/programs/）。

3.1 调用流程

第一步：获取网络上下文

客户端：

```
//创建阻塞的网络上下文
mbedtls_sock*netfd = mbedtls_socket(0);
```

服务器：

```
//创建阻塞的网络上下文
mbedtls_sock remote_fd;
mbedtls_sock*local_fd = mbedtls_socket(0);
```

第二步：创建并初始化 tls 上下文

客户端：

```
//创建并初始化 tls 客户端上下文
mbedtls_context *pContext = (mbedtls_context *)mbedtls_init_context(0);
```

服务器：

```
//创建并初始化 tls 服务器上下文
mbedtls_context *pContext = (mbedtls_context *)mbedtls_init_context(1);
```

第三步： 配置 tls 上下文

客户端：

```
//配置 tls 客户端上下文
if ((ret = mbedtls_config_context(pContext, (void *) &client_param,
                                  MBEDTLS_SSL_CLIENT_VERIFY_LEVEL)) != 0) {
       HC_ERR(("https: config failed.."));
       return -1;
```



```
}
```

服务器：

```
//配置 tls 客户端上下文
mbedtls_config_context(ssl_context,  (void *) &server_param,
MBEDTLS_SSL_SERVER_VERIFY_LEVEL);
```

注意：1.客户端与服务器所需要的证书参数是不同的，请参考具体代码。

2.客户端与服务器的证书验证模式通常是不同的，服务器基本上无需验证客户端的证书。

第四步： 建立连接

客户端：

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

服务器：

```
//等待客户端连接
ret = mbedtls_accept(mbedtls_context *context, mbedtls_sock *local_fd,
                     &remote_fd);
```

第五步：协商握手

```
//执行握手流程
if ((ret = mbedtls_handshake(g_pContext, &g_httpc_net_fd)) != 0)
       return -1;
```

第六步：发送/接收数据

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

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


第七步：释放 tls 上下文

```
//释放 tls 上下文
mbedtls_deinit_context(g_pContext);
```

第八步：释放网络上下文

客户端：

```
//释放网络上下文
mbedtls_closesocket(net_fd);
```

服务器：

```
//释放网络上下文
mbedtls_closesocket(local_fd);
mbedtls_closesocket(remote_fd);
```


4 其他配置

1. 通过 Makefile 中配置头文件

   config-xr-mini-cli.h　　//支持 client

   config-xr-mini-cliserv.h　　//支持 client/server

   config-xr-mini-serv.h　　//支持 server

   若上述三种配置均不能满足读者的要求，需要重新配置生成配置头文件。

2. 调试打印接口（位于配置头文件中）

   #define MBEDTLS_DEBUG_C 头文件打开该宏

3. 配置输入输出 buffer 大小（位于配置头文件中）

   #define MBEDTLS_SSL_MAX_CONTENT_LEN　　(6*1024)