



**Hi3861V100 / Hi3861LV100 TLS&DTLS**

## **开发指南**

文档版本 03

发布日期 2020-07-24

版权所有 © 上海海思技术有限公司2020。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

## 商标声明



**HISILICON**、海思和其他海思商标均为海思技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

## 注意

您购买的产品、服务或特性等应受海思公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，海思公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

# 上海海思技术有限公司

地址：            深圳市龙岗区坂田华为总部办公楼    邮编：518129

网址：            <https://www.hisilicon.com/cn/>

客户服务邮箱：  [support@hisilicon.com](mailto:support@hisilicon.com)



## 前言

### 概述

本文档主要介绍TLS/DTLS组件的开发实现示例。

TLS/DTLS以及其他加密套基于开源组件mbedtls 2.16.6实现，详细说明请参见官方说明：<https://tls.mbed.org/api/index.html>

如果官方说明版本与SDK版本不一致，请参考官方release说明：<https://github.com/ARMmbed/mbedtls/releases>

### 产品版本

与本文档相对应的产品版本如下。

产品名称	产品版本
Hi3861	V100
Hi3861L	V100


### 读者对象

本文档主要适用于以下工程师：




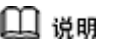
- 技术支持工程师
- 软件开发工程师

### 符号约定

在本文中可能出现下列标志，它们所代表的含义如下。

符号	说明
 危险	表示如不可避免则将会导致死亡或严重伤害的具有高等级风险的危害。



符号	说明
 <b>警告</b>	表示如不可避免则可能导致死亡或严重伤害的具有中等级风险的危害。
 <b>注意</b>	表示如不可避免则可能导致轻微或中度伤害的具有低等级风险的危害。
 <b>须知</b>	用于传递设备或环境安全警示信息。如不可避免则可能会导致设备损坏、数据丢失、设备性能降低或其它不可预知的结果。 “须知”不涉及人身伤害。
 <b>说明</b>	对正文中重点信息的补充说明。 “说明”不是安全警示信息，不涉及人身、设备及环境伤害信息。

## 修改记录

文档版本	发布日期	修改说明
03	2020-07-24	<ul style="list-style-type: none"><li>在“<a href="#">1 API接口说明</a>”中删除mbedtls的版本号。</li><li>在“<a href="#">2 开发指南</a>”中删除mbedtls的版本号。</li><li>在“<a href="#">3.2 配置说明</a>”中更新适配硬件算法加速器的内容。</li><li>在“<a href="#">3.3 适配说明</a>”中更新<a href="#">3.3.3 大数模幂适配</a>的标题和内容；更新<a href="#">3.3.4 随机数适配</a>的内容；新增<a href="#">3.3.5 CCM适配</a>小节；新增<a href="#">3.3.6 RSA数字签名适配</a>小节；新增<a href="#">3.3.7 ECDSA数字签名适配</a>小节；新增<a href="#">3.3.8 ECP适配</a>小节。</li></ul>
02	2020-07-21	更新“ <a href="#">概述</a> ”中mbedtls的版本号为mbedtls 2.16.6。
01	2020-04-30	第一次正式版本发布。
00B01	2020-01-15	第一次临时版本发布。



## 目录

前言.....	i
1 API 接口说明.....	1
1.1 结构体说明.....	1
1.2 API 列表.....	1
1.3 配置说明.....	1
2 开发指南.....	2
3 硬件适配.....	3
3.1 结构体说明.....	3
3.2 配置说明.....	3
3.3 适配说明.....	4
3.3.1 AES 适配.....	4
3.3.2 SHA256 适配.....	4
3.3.3 大数模幂适配.....	4
3.3.4 随机数适配.....	4
3.3.5 CCM 适配.....	4
3.3.6 RSA 数字签名适配.....	5
3.3.7 ECDSA 数字签名适配.....	5
3.3.8 ECP 适配.....	5
4 注意事项.....	6
4.1 关于 MBEDTLS_SHA256_ALT 的注意事项.....	6



# 1 API 接口说明

---

[1.1 结构体说明](#)

[1.2 API列表](#)

[1.3 配置说明](#)

## 1.1 结构体说明

mbbedtls详细的结构体说明请参考官方说明文档：<https://tls.mbed.org/api/annotated.html>

## 1.2 API 列表

mbbedtls详细的API说明请参考官方说明文档：[https://tls.mbed.org/api/globals\\_func.html](https://tls.mbed.org/api/globals_func.html)

## 1.3 配置说明

mbbedtls详细的配置项说明请参考官方说明文档：[https://tls.mbed.org/api/config\\_8h.html#ab3bca0048342cf2789e7d170548ff3a5](https://tls.mbed.org/api/config_8h.html#ab3bca0048342cf2789e7d170548ff3a5)



# 2 开发指南

---

mbedtls详细的开发DEMO请参考官方说明文档: <https://tls.mbed.org/api/modules.html>



# 3 硬件适配

## 3.1 结构体说明

## 3.2 配置说明

## 3.3 适配说明

## 3.1 结构体说明

- 在开启MBEDTLS\_AES\_ALT后，mbedtls\_aes\_context结构体定义被重定义为hi\_cipher\_aes\_ctrl，以下为结构体说明，更多内容请参考hi\_cipher.h。

```
typedef struct {
    hi_u32 key[AES_MAX_KEY_IN_WORD]; /* Key input. */
    hi_u32 iv[AES_IV_LEN_IN_WORD]; /* Initialization vector (IV). */
    hi_bool random_en; /* Enable random delay or not. */
    hi_cipher_aes_key_from key_from; /* Key from, When using kdf key, no need to
    configure the input key. */
    hi_cipher_aes_work_mode work_mode; /* Work mode. */
    hi_cipher_aes_key_length key_len; /* Key length. aes-ecb/cbc/ctr support 128/192/256
    bits key, ccm just support
                                128 bits key, xts just support 256/512 bits key. */
    hi_cipher_aes_ccm *ccm; /* Struct for ccm. */
}hi_cipher_aes_ctrl;
```
- 在开启MBEDTLS\_SHA256\_ALT后，mbedtls\_sha256\_context结构体定义被重定义为hi\_cipher\_hash\_atts，以下为结构体说明，更多内容请参考hi\_cipher.h。

```
typedef struct {
    const hi_u8 *hmac_key; /* hmac_key, just used for hmac. */
    hi_u32 hmac_key_len; /* hmac_key_len, just used for hmac. */
    hi_cipher_hash_type sha_type; /* sha_type, hash or hmac type. */
}hi_cipher_hash_atts;
```

## 3.2 配置说明

工程中默认使能以下配置，适配硬件算法加速器：

- MBEDTLS\_AES\_ALT
- MBEDTLS\_ENTROPY\_HARDWARE\_ALT
- MBEDTLS\_ECP\_ALT





- MBEDTLS\_RSA\_HW\_ACCEL\_BY\_HI\_CIPHER
- MBEDTLS\_ECDSA\_HW\_ACCEL\_BY\_HI\_CIPHER

除了以上编译宏控制的硬件加速外，mbedtls也对CCM进行加密和解密操作，大数模幂操作进行了硬件加速。这两个加速适配在发布版本中是默认开启的，目前暂无特定的编译宏进行控制。

还可以使能以下配置，适配额外的硬件加速器：

- MBEDTLS\_SHA256\_ALT

## 3.3 适配说明

### 3.3.1 AES 适配

- 使能MBEDTLS\_AES\_ALT后，ECB和CBC模式的AES算法会直接调用硬件驱动接口，而其他加密模式会沿用软件逻辑，最终调用硬件提供的AES\_ECB算法完成加速。
- 使能MBEDTLS\_AES\_ALT后，AES算法在使用硬件加速器时，会锁定硬件加速器资源，即AES操作是阻塞的，直至驱动获取资源或超时返回失败。

### 3.3.2 SHA256 适配

- 使能MBEDTLS\_SHA256\_ALT后，SHA256将使用硬件驱动接口，将不再支持SHA224操作。
- 使能MBEDTLS\_SHA256\_ALT后，SHA256算法在使用硬件加速器时，会锁定硬件加速器资源，即SHA256操作是阻塞的，直至驱动获取资源或超时返回失败。

### 3.3.3 大数模幂适配

- 发布版本中默认已对大数模幂操作（API接口为：mbedtls\_mpi\_exp\_mod）完成硬件适配。目前仅支持4096Bit（含）以下的大数模幂操作，超过4096Bit时，大数模幂操作失败，返回值为MBEDTLS\_ERR\_MPI\_BAD\_INPUT\_DATA。
- 硬件加速大数模幂操作时，会根据大数参数的位数，额外从堆中分配1KB~4KB的内存，如果分配失败，大数模幂操作会失败，返回值为MBEDTLS\_ERR\_MPI\_ALLOC\_FAILED。

### 3.3.4 随机数适配

使能MBEDTLS\_ENTROPY\_HARDWARE\_ALT后，系统会选用默认增加硬件随机数作为一个强随机数源，用户仍然可以增加额外的随机数源。

MBEDTLS\_ENTROPY\_HARDWARE\_ALT编译宏在发布版本中是默认开启的，除非用户明确已经注册其他强随机数源，否则此编译宏不允许被关闭。如果此宏被关闭，而用户也没有注册其他强随机数源，会导致mbedtls无法提供安全随机数，严重影响系统的安全性。

### 3.3.5 CCM 适配

对于加密方式为AES，密钥长度为128Bit的CCM的加密操作和解密操作，mbedtls已实现硬件适配，可以提升CCM加密与解密的速度。



- CCM加密的API接口为：mbedtls\_ccm\_encrypt\_and\_tag。
- CCM解密的API接口为：mbedtls\_ccm\_auth\_decrypt。

### 3.3.6 RSA 数字签名适配

config.h文件中使能MBEDTLS\_RSA\_HW\_ACCEL\_BY\_HI\_CIPHER编译宏之后，mbedtls会对密钥长度为2048Bit与4096Bit的RSA数字签名操作和验签操作进行硬件加速。对于密钥长度为非2048Bit或者非4096Bit的RSA数字签名与验签，仍然使用软件完成。

- RSA数字签名操作的API接口为：mbedtls\_rsa\_pkcs1\_sign。
- RSA数字验签操作的API接口为：mbedtls\_rsa\_pkcs1\_verify。

MBEDTLS\_RSA\_HW\_ACCEL\_BY\_HI\_CIPHER编译宏在发布版本中是默认开启的，如果用户希望节省代码体积，可以关闭此宏，但是关闭后会降低2048Bit与4096Bit的RSA数字签名操作和验签操作的速度。

### 3.3.7 ECDSA 数字签名适配

config.h文件中使能MBEDTLS\_ECDSA\_HW\_ACCEL\_BY\_HI\_CIPHER编译宏后，mbedtls会对密钥长度为256Bit的ECDSA数字签名操作和验签操作进行硬件加速。对于密钥长度为非256Bit的ECDSA数字签名与验签，仍然使用软件完成。

- ECDSA数字签名操作的API接口为：mbedtls\_ecdsa\_sign。
- ECDSA数字验签操作的API接口为：mbedtls\_ecdsa\_verify。

MBEDTLS\_ECDSA\_HW\_ACCEL\_BY\_HI\_CIPHER编译宏在发布版本中是默认开启的，如果用户希望节省代码体积，可以关闭此宏，但是关闭后会降低256Bit的ECDSA数字签名与验签的速度。

### 3.3.8 ECP 适配

config.h文件中使能MBEDTLS\_ECP\_ALT编译宏后，mbedtls会对256Bit ECP曲线的点加操作进行硬件加速。

- 256Bit ECP曲线的点加操作的API接口为：mbedtls\_ecp\_muladd\_restartable。

这里所说的256Bit ECP曲线是指mbedtls本身支持的三条ECP曲线，分别是：

- SECP256R1
- SECP256K1
- Brainpool256R1如果用户需要使用

如果用户需要使用非256Bit ECP曲线，则会关闭ECP曲线点加的硬件加速。

MBEDTLS\_ECP\_ALT编译宏在发布版本中是默认开启的。如果用户希望节省代码体积，可以关闭此宏。但是关闭后会严重的降低256Bit ECP点加操作的速度。WiFi SAE协议会使用256Bit ECP点加，如果关闭MBEDTLS\_ECP\_ALT编译宏，会大幅增加SAE协议的时间，导致WiFi Mesh连接速度或者WPA3安全连接速度的降低。



# 4 注意事项

## 4.1 关于MBEDTLS\_SHA256\_ALT的注意事项

### 4.1 关于 MBEDTLS\_SHA256\_ALT 的注意事项

使能MBEDTLS\_SHA256\_ALT后，在使用时需要注意以下细节：

- SHA256 HASH或HMAC操作不能嵌套，即必须每次执行完当前SHA256 HASH/HMAC操作后，再进行下一次操作。

例如：以下操作在ctxA计算完成前，执行ctxB计算，ctxA在步骤7开始返回失败，并且无法得到预期的结果。

```
mbbedtls_sha256_context *ctxA, *ctxB;
unsigned char *inputA1, *inputA2, *inputB1, *inputB2, *outputA, *outputB;
...
1: mbedtls_sha256_starts_ret(&ctxA);
2: mbedtls_sha256_update_ret(inputA1, 64);
3: mbedtls_sha256_starts_ret(&ctxB);
4: mbedtls_sha256_update_ret(inputB1, 64);
5: mbedtls_sha256_update_ret(inputB2, 64);
6: mbedtls_sha256_finish_ret(outputB, 32);
7: mbedtls_sha256_update_ret(inputA2, 64);
8: mbedtls_sha256_finish_ret(outputA, 32);
```

- 同时使能MBEDTLS\_SHA256\_ALT和MBEDTLS\_AES\_ALT时，SHA256 HASH或HMAC操作不能和AES操作嵌套，即必须每次执行完当前SHA256 HASH/HMAC操作后，再进行AES操作。

例如：以下操作在ctxA计算完成前，执行AES计算，最终结果为AES计算结果正确，ctxA在步骤5开始返回失败，并且无法得到预期的结果。

```
mbbedtls_sha256_context *ctxA;
unsigned char *inputA1, *inputA2, *outputA, *inputDec, *outputDec;
mbbedtls_aes_context *ctxDec = aes_decrypt_init(aesKey, 16);
...
1: mbedtls_sha256_starts_ret(&ctxA);
2: mbedtls_sha256_update_ret(inputA1, 64);
3: aes_decrypt(ctxDec, inputDec, outputDec);
4: aes_decrypt_deinit(ctxDec);
5: mbedtls_sha256_update_ret(inputA2, 64);
6: mbedtls_sha256_finish_ret(outputA, 32);
```



- 使能MBEDTLS\_SHA256\_ALT后，将不支持SHA224，使用SHA224时会返回失败。