



**Hi3861V100 / Hi3861LV100 TLS&DTLS**

## **开发指南**

文档版本 01

发布日期 2020-04-30

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 2.16.2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
 警告	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。
 注意	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。
须知	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。
 说明	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
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 DHM 适配.....	4
3.3.4 随机数适配.....	4
4 注意事项.....	5
4.1 关于 MBEDTLS_SHA256_ALT 的注意事项.....	5



# 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

MBEDTLS 2.16.2 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

MBEDTLS 2.16.2 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

MBEDTLS 2.16.2 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 2.16.2 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\_DHM\_ALT
- MBEDTLS\_ENTROPY\_HARDWARE\_ALT





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 DHM 适配

- 使能MBEDTLS\_DHM\_ALT后，mbedtls\_dhm\_make\_params中的模幂算子会根据传入的参数位数，选择使用软件实现或硬件加速，当所有大数参数都在4096bit（含）以下时，会采用硬件算法加速，超过4096bit时，采用软件实现。
- 使用硬件加速时，会根据大数参数位数，额外从堆中申请1KB~4KB的空间，如果空间不足会返回失败。

### 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。



# 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。