



**Hi3861 V100 / Hi3861L V100 TLS&DTLS**

## **Development Guide**

<b>Issue</b>	<b>01</b>
<b>Date</b>	<b>2020-04-30</b>

**Copyright © HiSilicon (Shanghai) Technologies Co., Ltd. 2020. All rights reserved.**

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of HiSilicon (Shanghai) Technologies Co., Ltd.

## **Trademarks and Permissions**



**HISILICON**, and other HiSilicon icons are trademarks of HiSilicon Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

## **Notice**

The purchased products, services and features are stipulated by the contract made between HiSilicon and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided “AS IS” without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

## **HiSilicon (Shanghai) Technologies Co., Ltd.**

Address: New R&D Center, 49 Wuhe Road,  
Bantian, Longgang District,  
Shenzhen 518129 P. R. China

Website: <https://www.hisilicon.com/en/>

Email: [support@hisilicon.com](mailto:support@hisilicon.com)



---

# About This Document

---

## Purpose

This document describes the development and implementation examples of the TLS/DTLS component.

TLS/DTLS and other cipher suites are implemented based on the open-source component mbedtls 2.16.2. For details, see <https://tls.mbed.org/api/index.html>.

In case of any description discrepancy or inconsistency between the ARM release and this SDK, <https://github.com/ARMmbed/mbedtls/releases> shall prevail.

## Related Versions

The following table lists the product versions related to this document.

Product Name	Version
Hi3861	V100
Hi3861L	V100

## Intended Audience






The document is intended for:

- Technical support engineers
- Software development engineers

## Symbol Conventions

The symbols that may be found in this document are defined as follows.



Symbol	Description
 DANGER	Indicates a hazard with a high level of risk which, if not avoided, will result in death or serious injury.
 WARNING	Indicates a hazard with a medium level of risk which, if not avoided, could result in death or serious injury.
 CAUTION	Indicates a hazard with a low level of risk which, if not avoided, could result in minor or moderate injury.
 NOTICE	Indicates a potentially hazardous situation which, if not avoided, could result in equipment damage, data loss, performance deterioration, or unanticipated results. NOTICE is used to address practices not related to personal injury.
 NOTE	Supplements the important information in the main text. NOTE is used to address information not related to personal injury, equipment damage, and environment deterioration.

## Change History

Issue	Date	Change Description
01	2020-04-30	This issue is the first official release.
00B01	2020-01-15	This issue is the first draft release.



# Contents

<b>About This Document.....</b>	<b>i</b>
<b>1 APIs.....</b>	<b>1</b>
1.1 Data Structures.....	1
1.2 APIs.....	1
1.3 Configuration.....	1
<b>2 Development Guidance.....</b>	<b>2</b>
<b>3 Hardware Adaptation.....</b>	<b>3</b>
3.1 Data Structures.....	3
3.2 Configuration.....	3
3.3 Adaptation Description.....	4
3.3.1 AES Adaptation.....	4
3.3.2 SHA256 Adaptation.....	4
3.3.3 DHM Adaptation.....	4
3.3.4 Random Number Adaptation.....	4
<b>4 Precautions.....</b>	<b>5</b>
4.1 Precautions for MBEDTLS_SHA256_ALT.....	5



# 1 APIs

---

[1.1 Data Structures](#)

[1.2 APIs](#)

[1.3 Configuration](#)

## 1.1 Data Structures

For details about the data structures of mbedtls 2.16.2, see <https://tls.mbed.org/api/annotated.html>.

## 1.2 APIs

For details about the APIs of mbedtls 2.16.2, see [https://tls.mbed.org/api/globals\\_func.html](https://tls.mbed.org/api/globals_func.html).

## 1.3 Configuration

For details about the configuration of mbedtls 2.16.2, see [https://tls.mbed.org/api/config\\_8h.html#ab3bca0048342cf2789e7d170548ff3a5](https://tls.mbed.org/api/config_8h.html#ab3bca0048342cf2789e7d170548ff3a5).



# 2 Development Guidance

---

For details about the development demo of mbedtls 2.16.2, see [https://  
tls.mbed.org/api/modules.html](https://tls.mbed.org/api/modules.html).



# 3 Hardware Adaptation

## 3.1 Data Structures

### 3.2 Configuration

### 3.3 Adaptation Description

## 3.1 Data Structures

- After **MBEDTLS\_AES\_ALT** is enabled, the **mbedtls\_aes\_context** structure is redefined as **hi\_cipher\_aes\_ctrl**. See the following code. For details, see **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;
```

- After **MBEDTLS\_SHA256\_ALT** is enabled, the **mbedtls\_sha256\_context** structure is redefined as **hi\_cipher\_hash\_atts**. See the following code. For details, see **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 Configuration

By default, the following options are enabled in the project to adapt to the hardware algorithm accelerators:

- MBEDTLS\_AES\_ALT**





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

You can also enable the following option to adapt to more hardware accelerators:

- MBEDTLS\_SHA256\_ALT

## 3.3 Adaptation Description

### 3.3.1 AES Adaptation

- After **MBEDTLS\_AES\_ALT** is enabled, AES-ECB and AES-CBC will directly call the hardware driver APIs. For other encryption modes, the software logic will be used. Finally, the AES\_ECB algorithm provided by the hardware will be used to complete acceleration.
- After **MBEDTLS\_AES\_ALT** is enabled, the AES algorithm locks the hardware accelerator resources when using the hardware accelerator. That is, the AES operation is blocking until the driver obtains resources or a failure message is returned due to timeout.

### 3.3.2 SHA256 Adaptation

- After **MBEDTLS\_SHA256\_ALT** is enabled, SHA256 will use hardware driver APIs. SHA224 operations are not supported.
- After **MBEDTLS\_SHA256\_ALT** is enabled, the SHA256 algorithm locks the hardware accelerator resources when using the hardware accelerator. That is, the SHA256 operation is blocking until the driver obtains resources or a failure message is returned due to timeout.

### 3.3.3 DHM Adaptation

- After **MBEDTLS\_DHM\_ALT** is enabled, the modular exponentiation operator in **mbedtls\_dhm\_make\_params** selects from software implementation and hardware acceleration based on the number of passed parameter bits. If the bit length is less than or equal to 4096 bits, the hardware acceleration is selected. If the bit length exceeds 4096 bits, software implementation is used.
- In hardware acceleration mode, 1 KB to 4 KB space is allocated from the heap based on the number of bits of the big number arguments. If the space is insufficient, a failure message is returned.

### 3.3.4 Random Number Adaptation

After **MBEDTLS\_ENTROPY\_HARDWARE\_ALT** is enabled, the system adds the default hardware random number as the strong random number source. You can add additional random number sources as required.



# 4 Precautions

## 4.1 Precautions for MBEDTLS\_SHA256\_ALT

### 4.1 Precautions for MBEDTLS\_SHA256\_ALT

After **MBEDTLS\_SHA256\_ALT** is enabled, pay attention to the following details:

- The SHA256 hash and HMAC operations must not be nested. That is, the next operation can be performed only after the current SHA256 hash or HMAC operation is complete.

For example, before the ctxA computation is complete, the ctxB computation is performed. The ctxA returns a failure message at step 7 and the expected result cannot be obtained.

```
mbdttls_sha256_context *ctxA, *ctxB;
unsigned char *inputA1, *inputA2, *inputB1, *inputB2, *outputA, *outputB;
...
1: mbdttls_sha256_starts_ret(&ctxA);
2: mbdttls_sha256_update_ret(inputA1, 64);
3: mbdttls_sha256_starts_ret(&ctxB);
4: mbdttls_sha256_update_ret(inputB1, 64);
5: mbdttls_sha256_update_ret(inputB2, 64);
6: mbdttls_sha256_finish_ret(outputB, 32);
7: mbdttls_sha256_update_ret(inputA2, 64);
8: mbdttls_sha256_finish_ret(outputA, 32);
```

- When both **MBEDTLS\_SHA256\_ALT** and **MBEDTLS\_AES\_ALT** are enabled, the SHA256 hash or HMAC operation cannot be nested with the AES operation. That is, the AES operation can be performed only after the current SHA256 hash or HMAC operation is complete.

For example, the following operations are performed before the ctxA calculation is complete. The final result is that the AES calculation result is correct. The ctxA returns a failure at step 5 and the expected result cannot be obtained.

```
mbdttls_sha256_context *ctxA;
unsigned char *inputA1, *inputA2, *outputA, *inputDec, *outputDec;
mbdttls_aes_context *ctxDec = aes_decrypt_init(aesKey, 16);
...
1: mbdttls_sha256_starts_ret(&ctxA);
2: mbdttls_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);
```

- After **MBEDTLS\_SHA256\_ALT** is enabled, SHA224 is not supported. A failure message is returned at an attempt to use SHA224.