# GmToolBox on FreeRTOS & RISC-V Documentation

# GmToolBox on FreeRTOS & RISC-V

# 技术文档

北理工的恶龙

2022-05-31

# 关于本文档

本技术文档为用户提供：

1. GmToolBox on FreeRTOS & RISC-V 项目介绍。
2. 利用 GmToolBox on FreeRTOS & RISC-V 开发时的编程指南。

# 发布说明

| 日期 | 版本 | 发布说明 |
| --- | --- | --- |
| 2022-05-31 | V1.0 | 初始版本 |

# 目录

# 1. 关于本项目

## 1.1 项目简介

本项目基于GmSSL开源项目开发，将其部分主要功能（SM3 / SM4 国密散列/加密算法）移植到基于RISC-V的FreeRTOS实时操作系统上。并针对部分应用场景进行了功能拓展和优化。

## 1.2 项目特性

本项目相较于国密算法的简单实现，具有以下特性：

- 执行效率高：在 代码层面充分优化 ，利用循环展开等技术，提高软件层面运行效率
- 针对嵌入式优化： 考虑到嵌入式设备的特性 （如存储空间小、运行内存小、处理器频率低等），优化功能结构，并在规模上加以限制，保障本库在嵌入式设备实体平台上运行的安全性。
- 模块化：功能模块耦合度低，模块复用度高、逻辑清晰
- 功能丰富：在实现SM3、SM4主算法的基础上，为了方便不同的散列、加密方式使用，提供了种类丰富的加密方式接口。使用者不需要复杂编码即可实现大部分加密模式。

# 2. SM3 散列算法工具

## 2.1 概述

SM3散列算法工具提供基于SM3散列算法的系列加密工具。

## 2.2 功能描述

支持SM3散列算法的哈希计算。

## 2.3 API 参考

对应的头文件 `sm3.h`

为用户提供以下接口：

- sm3_digest
- sm3_stream

### 2.3.1 sm3_digest

#### 2.3.1.1 描述

基于内存数据的 SM3 散列值计算。

#### 2.3.1.2 函数原型

```
void sm3_digest(const uint8_t *data, size_t datalen, uint8_t
dgst[SM3_DIGEST_SIZE]);
```

#### 2.3.1.3 参数

| 参数名称 | 类型 | 描述 | 输入输出 |
| --- | --- | --- | --- |
| data | uint8_t* | 待 SM3 计算的数据指针 | 输入 |
| datalen | size_t | 待 SM3 计算的数据长度（字节） | 输入 |
| dgst | uint8_t [] | SM3 散列计算的输出结果 | 输出 |

#### 2.3.1.4 举例

```
uint8_t sm3_res[SM3_DIGEST_SIZE];
sm3_digest((uint8_t*)"This is a test.", 15, sm3_res);
```

### 2.3.2 sm3_stream

#### 2.3.2.1 描述

基于缓冲区/文件描述符数据的 SM3 散列值流式计算。

#### 2.3.2.2 函数原型

```
int sm3_stream(uint32_t msglen, uint8_t dgst[SM3_DIGEST_SIZE], FILE
*fin);
```

#### 2.3.2.3 参数

| 参数名称 | 类型 | 描述 | 输入输出 |
| --- | --- | --- | --- |
| msglen | uint32_t | 待 SM3 计算的数据长度（字节） | 输入 |
| dgst | uint8_t [] | SM3 散列计算的输出结果 | 输出 |
| fin | FILE* | 待 SM3 计算的数据来源文件描述符 | 输入 |

#### 2.3.2.4 举例

```
uint8_t sm3_res[SM3_DIGEST_SIZE];
sm3_stream(100, sm3_res, stdin);
```

# 3. SM4 对称加密算法工具

## 3.1 概述

SM4对称加密算法工具提供基于SM4对称加密算法的系列加密工具。

## 3.2 功能描述

支持SM4对称加密算法的多种加密模式的加密计算。

## 3.3 API 参考

对应的头文件 `sm4.h`

为用户提供以下接口：

- sm4_set_encrypt_key
- sm4_set_decrypt_key
- sm4_cbc_encrypt_init
- sm4_cbc_encrypt_update
- sm4_cbc_encrypt_finish
- sm4_cbc_decrypt_init
- sm4_cbc_decrypt_update
- sm4_cbc_decrypt_finish
- sm4_ctr_encrypt_init
- sm4_ctr_encrypt_update
- sm4_ctr_encrypt_finish
- sm4_ctr_decrypt_init
- sm4_ctr_decrypt_update
- sm4_ctr_decrypt_finish
- sm4_gcm_encrypt
- sm4_gcm_decrypt

### 3.3.1 sm4_set_encrypt_key

#### 3.3.1.1 描述

将文本形式的加密密钥转换成特定类型的加密密钥。

#### 3.3.1.2 函数原型

```
void sm4_set_encrypt_key(SM4_KEY *key, const uint8_t
raw_key[SM4_KEY_SIZE]);
```

#### 3.3.1.3 参数

| 参数名称 | 类型 | 描述 | 输入输出 |
| --- | --- | --- | --- |
| key | SM4_KEY* | 转换后的加密密钥 | 输出 |
| raw_key | uint8_t[] | 未转换的加密密钥 | 输入 |

#### 3.3.1.4 举例

```
sm4_set_encrypt_key(&myKey, "This is test key");
```

### 3.3.2 sm4_set_decrypt_key

#### 3.3.2.1 描述

将文本形式的加密密钥转换成特定类型的加密密钥。

#### 3.3.2.2 函数原型

```
void sm4_set_decrypt_key(SM4_KEY *key, const uint8_t
raw_key[SM4_KEY_SIZE]);
```

#### 3.3.2.3 参数

| 参数名称 | 类型 | 描述 | 输入输出 |
| --- | --- | --- | --- |
| key | SM4_KEY* | 转换后的解密密钥 | 输出 |
| raw_key | uint8_t[] | 未转换的解密密钥 | 输入 |

#### 3.3.2.4 举例

```
sm4_set_decrypt_key(&myKey, "This is test key");
```

### 3.3.3 sm4_cbc_encrypt_init

#### 3.3.3.1 描述

CBC模式加密的初始化。

#### 3.3.3.2 函数原型

```
int sm4_cbc_encrypt_init(SM4_CBC_CTX *ctx, const uint8_t
key[SM4_KEY_SIZE], const uint8_t iv[SM4_BLOCK_SIZE]);
```

#### 3.3.3.3 参数

| 参数名称 | 类型 | 描述 | 输入输出 |
| --- | --- | --- | --- |
| ctx | SM4_CBC_CTX* | context | 输出 |
| key | uint8_t [] | 加密密钥 | 输入 |
| iv | uint8_t [] | 初始向量 | 输入 |

#### 3.3.3.4 举例

```
uint8_t iv[SM4_BLOCK_SIZE];
sm4_cbc_encrypt_init(&ctx, "This is test key",iv);
```

### 3.3.4 sm4_cbc_encrypt_update

#### 3.3.4.1 描述

输出完整的块的CBC模式加密结果

#### 3.3.4.2 函数原型

```
int sm4_cbc_encrypt_update(SM4_CBC_CTX *ctx, const uint8_t *in,
size_t inlen, uint8_t *out, size_t *outlen);
```

#### 3.3.4.3 参数

| 参数名称 | 类型 | 描述 | 输入输出 |
| --- | --- | --- | --- |
| ctx | SM4_CBC_CTX* | context | 输入 |
| in | uint8_t * | 待加密明文 | 输入 |
| inlen | size_t | 输入长度 | 输入 |
| out | uint8_t * | 加密结果 | 输出 |
| outlen | size_t* | 结果长度 | 输出 |

#### 3.3.4.4 举例

```
uint8_t iv[SM4_BLOCK_SIZE];
sm4_cbc_encrypt_update(&ctx, "This is test key",iv);
```

### 3.3.5 sm4_cbc_encrypt_finish

#### 3.3.5.1 描述

输出需要填充的块的CBC模式加密结果

#### 3.3.5.2 函数原型

```
int sm4_cbc_encrypt_finish(SM4_CBC_CTX *ctx, uint8_t *out, size_t
*outlen);
```

#### 3.3.5.3 参数

| 参数名称 | 类型 | 描述 | 输入输出 |
| --- | --- | --- | --- |
| ctx | SM4_CBC_CTX* | context | 输入 |
| out | uint8_t * | 加密结果 | 输出 |
| outlen | size_t* | 结果长度 | 输出 |

#### 3.3.5.4 举例

```
uint8_t iv[SM4_BLOCK_SIZE];
int sm4_cbc_encrypt_init(&ctx, "This is test key",iv);
```

### 3.3.6 sm4_cbc_decrypt_init

#### 3.3.6.1 描述

CBC模式解密的初始化。

#### 3.3.6.2 函数原型

```
int sm4_cbc_decrypt_init(SM4_CBC_CTX *ctx, const uint8_t
key[SM4_KEY_SIZE], const uint8_t iv[SM4_BLOCK_SIZE]);
```

#### 3.3.6.3 参数

| 参数名称 | 类型 | 描述 | 输入输出 |
| --- | --- | --- | --- |
| ctx | SM4_CBC_CTX* | context | 输出 |
| key | uint8_t [] | 加密密钥 | 输入 |
| iv | uint8_t [] | 初始向量 | 输入 |

#### 3.3.6.4 举例

```
uint8_t iv[SM4_BLOCK_SIZE];
sm4_cbc_decrypt_init(&ctx, "This is test key",iv);
```

### 3.3.7 sm4_cbc_decrypt_update

#### 3.3.7.1 描述

输出完整的块的CBC模式解密结果

#### 3.3.7.2 函数原型

```
int sm4_cbc_decrypt_update(SM4_CBC_CTX *ctx, const uint8_t *in,
size_t inlen, uint8_t *out, size_t *outlen);
```

#### 3.3.7.3 参数

| 参数名称 | 类型 | 描述 | 输入输出 |
| --- | --- | --- | --- |
| ctx | SM4_CBC_CTX* | context | 输入 |
| in | uint8_t * | 待解密密文 | 输入 |
| inlen | size_t | 输入长度 | 输入 |
| out | uint8_t * | 解密结果 | 输出 |
| outlen | size_t* | 结果长度 | 输出 |

#### 3.3.7.4 举例

```
uint8_t iv[SM4_BLOCK_SIZE];
sm4_cbc_encrypt_init(&ctx, "This is test key",iv);
```

### 3.3.8 sm4_cbc_decrypt_finish

#### 3.3.8.1 描述

输出需要填充的块的CBC模式解密结果

#### 3.3.8.2 函数原型

```
int sm4_cbc_decrypt_finish(SM4_CBC_CTX *ctx, uint8_t *out, size_t
*outlen);
```

#### 3.3.8.3 参数

| 参数名称 | 类型 | 描述 | 输入输出 |
| --- | --- | --- | --- |
| ctx | SM4_CBC_CTX* | context | 输入 |
| out | uint8_t * | 解密结果 | 输出 |
| outlen | size_t* | 结果长度 | 输出 |

#### 3.3.8.4 举例

```
uint8_t iv[SM4_BLOCK_SIZE];
int sm4_cbc_encrypt_init(&ctx, "This is test key",iv);
```

### 3.3.9 sm4_ctr_encrypt_init

#### 3.3.9.1 描述

CTR模式加密的初始化。

#### 3.3.9.2 函数原型

```
int sm4_ctr_encrypt_init(SM4_CTR_CTX *ctx, const uint8_t
key[SM4_KEY_SIZE], const uint8_t ctr[SM4_BLOCK_SIZE]);
```

#### 3.3.9.3 参数

| 参数名称 | 类型 | 描述 | 输入输出 |
| --- | --- | --- | --- |
| ctx | SM4_CTR_CTX* | context | 输出 |
| key | uint8_t [] | 加密密钥 | 输入 |
| ctr | uint8_t [] | 计数器 | 输入 |

#### 3.3.9.4 举例

```
uint8_t iv[SM4_BLOCK_SIZE];
sm4_cbc_encrypt_init(&ctx, "This is test key",iv);
```

### 3.3.10 sm4_ctr_encrypt_update

#### 3.3.10.1 描述

输出完整的块的CTR模式加密结果

#### 3.3.10.2 函数原型

```
int sm4_ctr_encrypt_update(SM4_CTR_CTX *ctx, const uint8_t *in,
size_t inlen, uint8_t *out, size_t *outlen);
```

#### 3.3.10.3 参数

| 参数名称 | 类型 | 描述 | 输入输出 |
| --- | --- | --- | --- |
| ctx | SM4_CTR_CTX* | context | 输入 |
| in | uint8_t * | 待加密明文 | 输入 |
| inlen | size_t | 输入长度 | 输入 |
| out | uint8_t * | 加密结果 | 输出 |
| outlen | size_t* | 结果长度 | 输出 |

#### 3.3.10.4 举例

```
uint8_t iv[SM4_BLOCK_SIZE];
sm4_cbc_encrypt_update(&ctx, "This is test key",iv);
```

### 3.3.11 sm4_ctr_encrypt_finish

#### 3.3.11.1 描述

输出需要填充的块的CTR模式加密结果

#### 3.3.11.2 函数原型

```
int sm4_ctr_encrypt_finish(SM4_CTR_CTX *ctx, uint8_t *out, size_t
*outlen);
```

#### 3.3.11.3 参数

| 参数名称 | 类型 | 描述 | 输入输出 |
| --- | --- | --- | --- |
| ctx | SM4_CTR_CTX* | context | 输入 |
| out | uint8_t * | 加密结果 | 输出 |
| outlen | size_t* | 结果长度 | 输出 |

#### 3.3.11.4 举例

```
uint8_t iv[SM4_BLOCK_SIZE];
int sm4_cbc_encrypt_init(&ctx, "This is test key",iv);
```

### 3.3.12 sm4_ctr_decrypt_init

#### 3.3.12.1 描述

CTR模式解密的初始化。

#### 3.3.12.2 函数原型

```
int sm4_ctr_decrypt_init(SM4_CTR_CTX *ctx, const uint8_t
key[SM4_KEY_SIZE], const uint8_t ctr[SM4_BLOCK_SIZE]);
```

#### 3.3.12.3 参数

| 参数名称 | 类型 | 描述 | 输入输出 |
| --- | --- | --- | --- |
| ctx | SM4_CTR_CTX* | context | 输出 |
| key | uint8_t [] | 加密密钥 | 输入 |
| ctr | uint8_t [] | 计数器 | 输入 |

#### 3.3.12.4 举例

```
uint8_t iv[SM4_BLOCK_SIZE];
sm4_cbc_encrypt_init(&ctx, "This is test key",iv);
```

### 3.3.13 sm4_ctr_decrypt_update

#### 3.3.13.1 描述

输出完整的块的CTR模式解密结果

#### 3.3.13.2 函数原型

```
int sm4_ctr_decrypt_update(SM4_CTR_CTX *ctx, const uint8_t *in,
size_t inlen, uint8_t *out, size_t *outlen);
```

#### 3.3.13.3 参数

| 参数名称 | 类型 | 描述 | 输入输出 |
| --- | --- | --- | --- |
| ctx | SM4_CTR_CTX* | context | 输入 |
| in | uint8_t * | 待加密明文 | 输入 |
| inlen | size_t | 输入长度 | 输入 |
| out | uint8_t * | 加密结果 | 输出 |
| outlen | size_t* | 结果长度 | 输出 |

#### 3.3.13.4 举例

```
uint8_t iv[SM4_BLOCK_SIZE];
sm4_cbc_encrypt_update(&ctx, "This is test key",iv);
```

### 3.3.14 sm4_ctr_decrypt_finish

#### 3.3.14.1 描述

输出需要填充的块的CTR模式解密结果

#### 3.3.14.2 函数原型

```
int sm4_ctr_decrypt_finish(SM4_CTR_CTX *ctx, uint8_t *out, size_t
*outlen);
```

#### 3.3.14.3 参数

| 参数名称 | 类型 | 描述 | 输入输出 |
| --- | --- | --- | --- |
| ctx | SM4_CTR_CTX* | context | 输入 |
| out | uint8_t * | 加密结果 | 输出 |
| outlen | size_t* | 结果长度 | 输出 |

#### 3.3.14.4 举例

```
uint8_t iv[SM4_BLOCK_SIZE];
int sm4_cbc_encrypt_init(&ctx, "This is test key",iv);
```

### 3.3.15 sm4_gcm_encrypt

#### 3.3.15.1 描述

GCM模式加密

#### 3.3.15.2 函数原型

```
int sm4_gcm_encrypt(const SM4_KEY *key, const uint8_t *iv, size_t
ivlen,
    const uint8_t *aad, size_t aadlen, const uint8_t *in, size_t
inlen,
    uint8_t *out, size_t taglen, uint8_t *tag);
```

#### 3.3.15.3 参数

| 参数名称 | 类型 | 描述 | 输入输出 |
| --- | --- | --- | --- |
| key | SM4_KEY* | 加密密钥 | 输入 |
| iv | uint8_t * | 初始向量 | 输入 |
| inlen | size_t | 初始向量长度 | 输入 |
| aad | uint8_t * | 附加认证数据 | 输入 |
| aadlen | size_t | 附加认证数据长度 | 输入 |
| in | uint8_t * | 待加密明文 | 输入 |
| inlen | size_t | 输入长度 | 输入 |
| out | uint8_t * | 加密结果 | 输出 |
| taglen | size_t | mac长度 | 输出 |
| tag | uint8_t * | mac | 输出 |

#### 3.3.15.4 举例

```
uint8_t iv[SM4_BLOCK_SIZE];
int sm4_cbc_encrypt_init(&ctx, "This is test key",iv);
```

### 3.3.16 sm4_gcm_decrypt

#### 3.3.16.1 描述

GCM模式解密

#### 3.3.16.2 函数原型

```
int sm4_gcm_decrypt(const SM4_KEY *key, const uint8_t *iv, size_t
ivlen,
    const uint8_t *aad, size_t aadlen, const uint8_t *in, size_t
inlen,
    const uint8_t *tag, size_t taglen, uint8_t *out);
```

#### 3.3.16.3 参数

| 参数名称 | 类型 | 描述 | 输入输出 |
| --- | --- | --- | --- |
| key | SM4_KEY* | 解密密钥 | 输入 |
| iv | uint8_t * | 初始向量 | 输入 |
| inlen | size_t | 初始向量长度 | 输入 |
| aad | uint8_t * | 附加认证数据 | 输入 |
| aadlen | size_t | 附加认证数据长度 | 输入 |
| in | uint8_t * | 待解密密文 | 输入 |
| inlen | size_t | 输入长度 | 输入 |
| tag | uint8_t * | mac | 输出 |
| taglen | size_t | mac长度 | 输出 |
| out | uint8_t * | 解密结果 | 输出 |

#### 3.3.16.4 举例

```
uint8_t iv[SM4_BLOCK_SIZE];
int sm4_cbc_encrypt_init(&ctx, "This is test key",iv);
```