M4521 ECC認證

NuMicro® 32位系列微控制器範例代碼介紹

文件資訊

應用簡述	於 M4521 開發板上透過 mbedtls 函數庫以純軟體實現 ECC 加解密運算功能
BSP 版本	M4521_BSP_CMSIS_V3.00.003
開發平台	NuTiny-SDK-M4521S

The information described in this document is the exclusive intellectual property of Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.

Nuvoton is providing this document only for reference purposes of NuMicro microcontroller and microprocessor based system design. Nuvoton assumes no responsibility for errors or omissions.

All data and specifications are subject to change without notice.

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

1. 概述

Mbedtls 是由 Mbed TLS 在 Apache 2.0 或 GPL 2.0 許可下以開源形式提供的 SSL(Secure Sockets Layer)函數庫，含有豐富的加解密函式可用來進行安全通訊，避免資料遭竊聽。

此範例程式利用 mbedtls 函數庫，以純軟體執行橢圓曲線密碼學(Elliptic Curve Cryptography, ECC)之金鑰加解密運算，以緩解部分開發板由於設計規格不同，而無法以硬體實現加解密功能的問題。

1.1 橢圓曲線密碼學

橢圓曲線密碼學是基於橢圓曲線的數學原理，屬對稱加密技術的公鑰加密體制。在同等的安全等級下，相較於其它加密演算法，ECC 具有更短的密鑰，也能在此基礎上提供更高的安全性等級加密，例如 256 bit 的 ECC 金鑰強度相當於 3072 bit 的 RSA 金鑰。ECC 數學邏輯更加複雜，故具有加解密較其它方法費時的缺點，但普遍認為 ECC 是給定密鑰長度情況下，最強大的非對稱加密演算法。

ECC 的數學基礎是利用橢圓曲線上的離散對數問題，針對密碼學應用的橢圓曲線是在有限域的平面曲線[1,2]，其中 ECC 最常使用的橢圓曲線方程式如下：

$$y^2 = x^3 + ax + b \quad (\mathrm{a,b} \in \mathrm{GF(p)}, 4a^3 + 27b^2 \neq 0)$$

其中 a 和 b 為任意實數，橢圓曲線會因應 a 和 b 的數值而在平面上呈不同形狀，但橢圓曲線始終會對稱於 x 軸，且方程式需滿足無奇異點，即圖形沒有尖點或自相交[3]，隨機橢圓曲線函數如圖 1-1、圖 1-2 所示。

1 https://zh.m.wikipedia.org/zh-tw/%E6%A4%AD%E5%9C%86%E6%9B%B2%E7%BA%BF%E5%AF%86%E7%A0%81%E5%AD%A6
2 https://zhuanlan.zhihu.com/p/42629724
3 https://zh.m.wikipedia.org/zh-tw/%E6%A4%AD%E5%9C%86%E6%9B%B2%E7%BA%BF

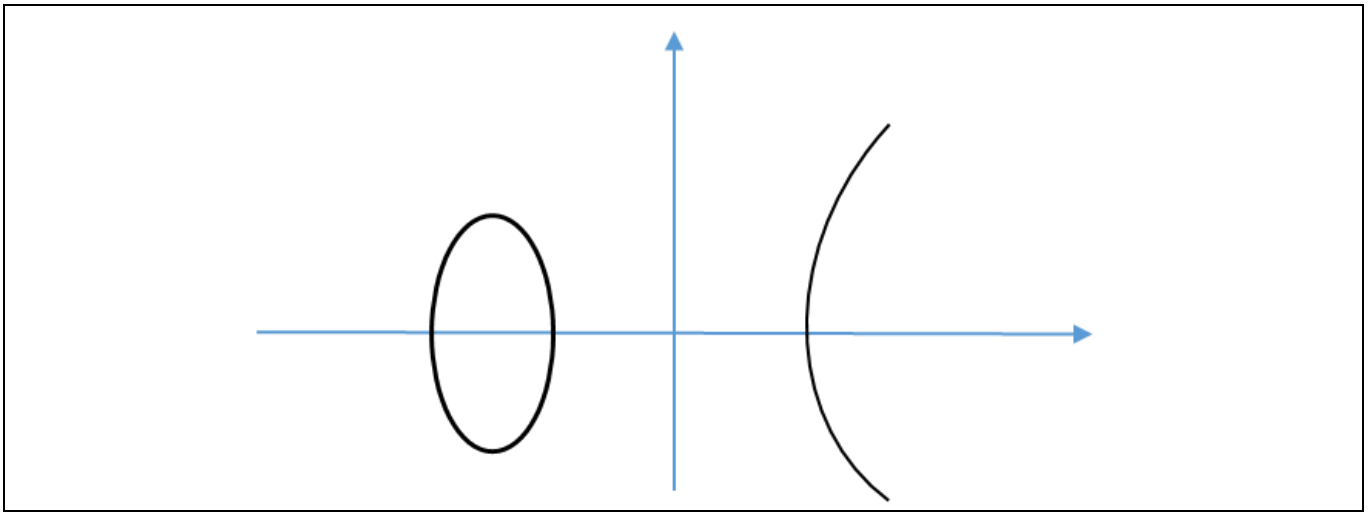

圖 1-1 橢圓曲線圖 ($y^2 = x^3 - 1$)

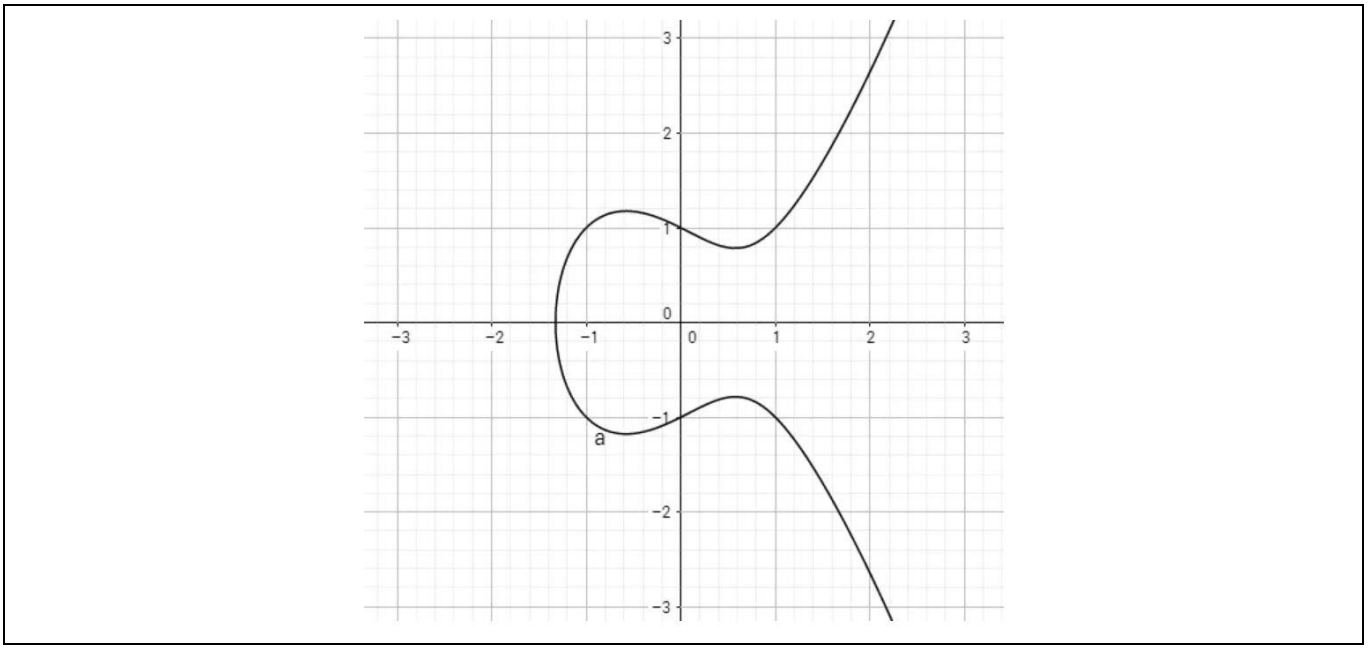

圖 1-2 橢圓曲線圖 2 ($y^2 = x^3 - x + 1$) [4]

4 https://www.jianshu.com/p/899077117ff8

1.2 橢圓曲線數位簽章演算法(ECDSA)

1.2.1 原理與操作

建立基於橢圓曲線的加密機制，需要找到類似 RSA 質因子分解或其他求離散對數這樣的難題。而橢圓曲線上的已知 G 和 xG 求 x，是非常困難的，此即為橢圓曲線上的的離散對數問題，因此橢圓曲線數位簽章(Elliptic Curve Digital Signature Algorithm, ECDAS)便由此而生。

在密碼學中會生成一對金鑰:公鑰(public key)與私鑰(private key)，相對橢圓曲線的加密機制，即在橢圓曲線上設 k 為私鑰，k=kG 為公鑰(G 為基點)，私鑰負責加密，並由發訊者保管；公鑰則對外公布，擁有者可用來解密對應資料。

當明文經簽章演算法產生摘要後，再以私鑰加密形成數位簽章才對外公布(非由私鑰直接加密明文)，擁有對應公鑰的使用者便能以公鑰將簽章解密，解出的摘要會與發送者加密前的內容一致，並藉此確認資料沒被竄改過。ECDSA 加解密流程如圖 1-3~圖 1-4 所示。

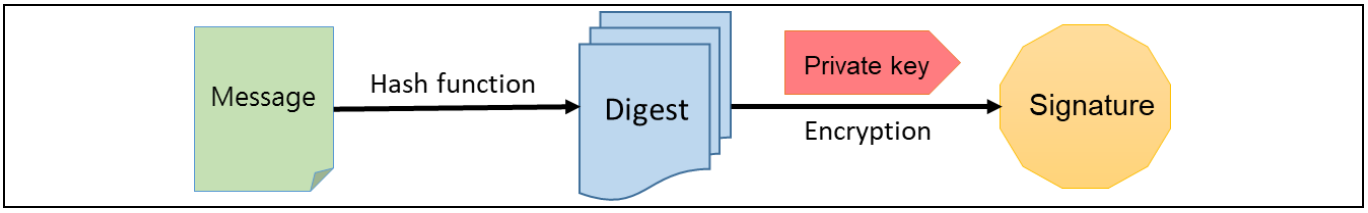

圖 1-3 私鑰加密形成簽章

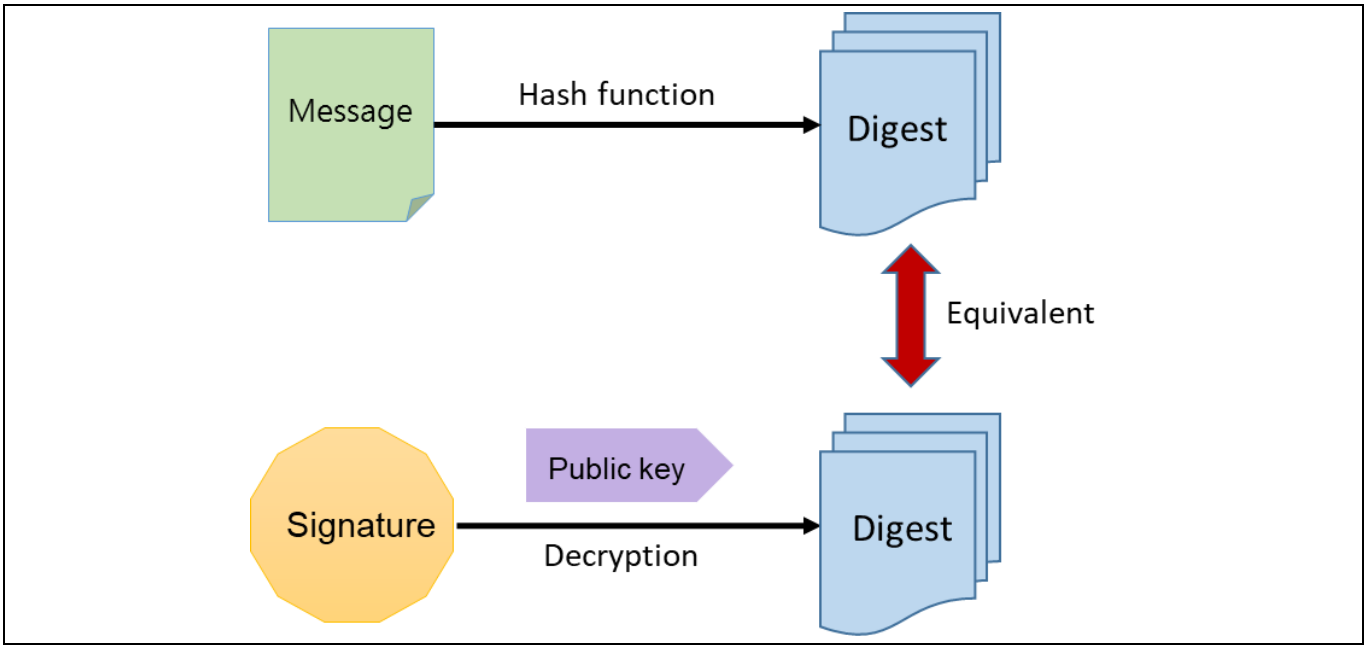

圖 1-4 公鑰對簽章解密

1.2.2 金鑰生成

首先依需求(位元數、簽章遵循標準等)選擇一組橢圓曲線，並由 mbedtls 函數庫呼叫 ***mbedtls_ecp_curve_info_from_grp_id()***取得該曲線相關資訊，用處理過的參數 gid 投入函數 ***mbedtls_ecdsa_genkey()*** 生成橢圓曲線上的金鑰對，最後再由 ***mbedtls_ecdsa_from_keypair()***基於金鑰對產生 ECDSA 金鑰參數。詳細函數說明如表 1-1~表 1-3 所示。

*mbedtls_ecp_curve_info_from_grp_id(mbedtls_ecp_group_id grp_id)		
參數：	grp_id	橢圓曲線參數
回傳值：		Null

表 1-1 橢圓曲線資訊擷取 API

mbedtls_ecdsa_genkey(mbedtls_ecdsa_context *ctx, mbedtls_ecp_group_id gid, int (*f_rng)(void *, unsigned char *, size_t), void *p_rng)		
參數：	*ctx	ECDSA金鑰參數
	gid	橢圓曲線參數
	*f_rng	亂數函數
	*p_rng	f_rng 附加函數 (不使用則填 NULL)
回傳值：		成功 = 0 失敗 = MBEDTLS_ERR_ECP_XXX

表 1-2 由橢圓曲線生成金鑰對 API

mbedtls_ecdsa_from_keypair(mbedtls_ecdsa_context *ctx, const mbedtls_ecp_keypair *key)		
參數：	*ctx	ECDSA金鑰參數
	*key	使用金鑰對
回傳值：		成功 = 0 失敗 = MBEDTLS_ERR_ECP_XXX

表 1-3 生成 ECDSA 金鑰參數 API

1.2.3 雜湊值(Hash value)

雜湊是電腦科學中一種對資料的處理方法，通過某種特定的函式/演算法(稱為雜湊函式/演算法)將要檢索的項與用來檢索的索引(稱為雜湊，或者雜湊值)關聯起來，生成一種便於搜尋的資料結構(稱為雜湊表)。它也常用作一種資訊安全的實作方法，由一串資料中經過雜湊演算法計算出來的資料指紋，經常用來識別檔案與資料是否有被竄改，以保證檔案與資料確實是由原創者所提供。雜湊演算法也被用來加密存在資料庫中的密碼字串，由於雜湊演算法所計算出來的雜湊值(Hash Value)具有不可逆(無法逆向演算回原本的數值)的性質，且雜湊值必須隨明文改變而改變，因此可有效的保護密碼。例如 SHA-256 是最廣泛使用的密碼雜湊函式之一，它生成 256 位元值的雜湊值[5]。

1.2.4 數位簽章

數位簽章的寫入與生成，是將雜湊函數處理後的訊息再以私鑰加密，並僅能以發送端釋出的公鑰做解密。可於 mbedtls 函數庫呼叫 ***mbedtls_ecdsa_write_signature()***做計算並寫入簽章，其中會以 ***mbedtls_ecdsa_write_signature_restartable()***逐步記錄已執行程序，遇到錯誤將立即回傳錯誤代碼，且可於指定位置執行重啟動作，以節省總阻塞時間(Total Blocking Time, TBT)。詳細函數說明如表 1-4~表 1-5 所示。

mbedtls_ecdsa_write_signature(mbedtls_ecdsa_context *ctx,mbedtls_md_type_t md_alg, const unsigned char *hash, size_t hlen, unsigned char *sig, size_t sig_size, size_t *slen, int (*f_rng)(void *, unsigned char *, size_t), void *p_rng)		
參數：	*ctx	ECDSA 金鑰參數
	md_alg	密碼雜湊函數
	*hash	原始雜湊值存放位址
	hlen	雜湊函數長度(Bytes)
	*sig	寫入簽章存位
	sig_size	簽章存位長度(Bytes)
	*slen	存儲簽名的實際長度的地址
	*f_rng	亂數函數

[5] https://zh.wikipedia.org/zh-tw/%E6%95%A3%E5%88%97%E5%87%BD%E6%95%B8

	*p_rng	f_rng 附加函數 (不使用則填 NULL)
回傳值：		成功 = 0 失敗 = MBEDTLS_ERR_ECP_XXX, MBEDTLS_ERR_MPI_XXX, MBEDTLS_ERR_ASN1_XXX

表 1-4 簽章寫入 API

mbedtls_ecdsa_write_signature_restartable(mbedtls_ecdsa_context *ctx, mbedtls_md_type_t md_alg, const unsigned char *hash, size_t hlen, unsigned char *sig, size_t sig_size, size_t *slen, int (*f_rng)(void *, unsigned char *, size_t), void *p_rng, mbedtls_ecdsa_restart_ctx *rs_ctx);		
參數：	*ctx	ECDSA 金鑰參數
	md_alg	密碼雜湊函數
	*hash	原始雜湊值存放位址
	hlen	雜湊函數長度(Bytes)
	*sig	寫入簽章存位
	sig_size	簽章存位長度(Bytes)
	*slen	存儲簽名的實際長度的地址
	*f_rng	亂數函數
	*p_rng	f_rng 附加函數 (不使用則填 NULL)
	*rs_ctx	重啟用的金鑰參數(不使用則填 NULL)
回傳值：		成功 = 0 超過最大操作次數 = MBEDTLS_ERR_ECP_IN_PROGRESS 失敗 = MBEDTLS_ERR_ECP_XXX, MBEDTLS_ERR_MPI_XXX, MBEDTLS_ERR_ASN1_XXX

表 1-5 簽章重啟 API

1.2.5 數位簽章驗證

由於數位簽章以發送端的私鑰處理資料，且僅能以發送端提供的公鑰通過驗證，故其常用於防止偽造、抵賴與檢測竄改。如圖 1-4 所示，數位簽章的認證重點為：公鑰解密後，所得摘要(Digest)是否與資料發送端將訊息經雜湊處理後所得的摘要一致，若兩者一致，代表簽章合法，反之為非法。可於 mbedtls 函數庫叫用 ***mbedtls_ecdsa_read_signature()***對數位簽章執行驗證，函數中也包含 ***mbedtls_ecdsa_write_signature_restartable()***進行輔助，詳細函數說明表 1-6~表 1-7 所示。

mbedtls_ecdsa_read_signature(mbedtls_ecdsa_context *ctx, const unsigned char *hash, size_t hlen, const unsigned char *sig, size_t slen)		
參數：	*ctx	ECDSA金鑰參數
	*hash	原始雜湊值存放位址
	hlen	雜湊函數長度(Bytes)
	*sig	寫入簽章存位
	slen	存儲簽名的實際長度的地址
回傳值：		成功 = 0 失敗 = MBEDTLS_ERR_ECP_XXX, MBEDTLS_ERR_MPI_XXX

表 1-6 簽章驗證 API

mbedtls_ecdsa_read_signature_restartable(mbedtls_ecdsa_context *ctx, const unsigned char *hash, size_t hlen, const unsigned char *sig, size_t slen, mbedtls_ecdsa_restart_ctx *rs_ctx)		
參數：	*ctx	ECDSA 金鑰參數
	*hash	原始雜湊值存放位址
	hlen	雜湊函數長度(Bytes)
	*sig	寫入簽章存位
	slen	存儲簽名的實際長度的地址
	*rs_ctx	重啟用的金鑰參數(不使用則填 NULL)
回傳值：		成功 = 0

	失敗 = MBEDTLS_ERR_ECP_XXX, MBEDTLS_ERR_MPI_XXX

表 1-7 重啟簽章驗證 API

1.3 執行結果

執行後會打印出以下資訊：

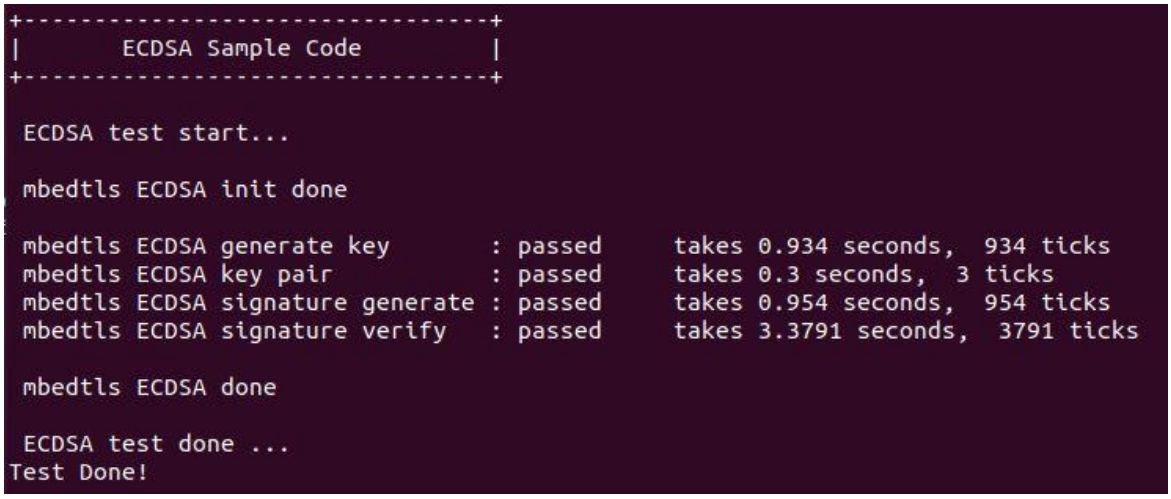

圖 1-5 執行結果

2. 代碼介紹

使用 mbedtls 函數庫產生 ECDSA 的公鑰及私鑰，代碼位於 *ecdsa_test.c*。

```
    mbedtls_ecdsa_context ecdsa;
    size_t sig_len;
    const mbedtls_ecp_curve_info *curve_info =
                                mbedtls_ecp_curve_info_from_grp_id(MBEDTLS_ECP_DP_BP256R1);
    mbedtls_pk_context pk;
    mbedtls_ecp_keypair *ecp;

    mbedtls_pk_init(&pk);

    /* Initializes an ECDSA context. */
    mbedtls_ecdsa_init(&ecdsa);
    printf(" mbedtls ECDSA init done  \n\n");

    printf(" mbedtls ECDSA generate key       : ");
    /* Generate the keypair on an elliptic curve */
    ret = mbedtls_ecdsa_genkey(&ecdsa, curve_info->grp_id, myrand, NULL);
    if(ret == 0) {
        printf("passed");
        u32Time = get_timer0_counter();
        printf("     takes %d.%d seconds,  %d ticks\n", u32Time / 1000000, u32Time / 1000,
g_u32Ticks);
    } else {
        printf("failed! ret[%d]\n", ret);
        mbedtls_ecdsa_free(&ecdsa);
        return ret;
    }

printf(" mbedtls ECDSA key pair           : ");
/* Generate the public key and private key from key pair */
    ret = mbedtls_ecdsa_from_keypair(&ecdsa, &ecdsa);
    if(ret == 0) {
        printf("passed");
        u32Time = get_timer0_counter();
        printf("     takes %d.%d seconds,  %d ticks\n", u32Time / 1000000, u32Time / 1000,
g_u32Ticks);
    } else {
        printf("failed! ret[%d]\n", ret);
        mbedtls_ecdsa_free(&ecdsa);
        return ret;
}
```

金鑰完成後，選用所需雜湊函數將資料加密，最後再以私鑰製作數位簽章。代碼位於 *ecdsa_test.c*。

```
/* Cacualate the data hash */
    const mbedtls_md_info_t *md_info;
    md_info = mbedtls_md_info_from_type(MBEDTLS_MD_SHA256);
    mbedtls_md(md_info,(const unsigned char *) test_image_bin, sizeof(test_image_bin),
hash);
```

```
    /* Computes the ECDSA signature and writes it to a buffer */
    printf(" mbedtls ECDSA signature generate : ");
    ret = mbedtls_ecdsa_write_signature(&ecdsa, MBEDTLS_MD_SHA256, hash, sizeof(hash),
tmp, sizeof(tmp), &sig_len, myrand, NULL);

    if(ret == 0) {
        printf("passed");
        u32Time = get_timer0_counter();
        printf("     takes %d.%d seconds,  %d ticks\n", u32Time / 1000000, u32Time / 1000,
g_u32Ticks);
    } else {
        printf("failed! ret[%d]\n", ret);
        mbedtls_ecdsa_free(&ecdsa);
        return ret;
    }
```

讀取數位簽章內容，並使用公鑰執行認證。代碼位於 *ecdsa_test.c*。

```
  /* Reads and verifies an ECDSA signature */
    printf(" mbedtls ECDSA signature verify   : ");
    ret = mbedtls_ecdsa_read_signature(&ecdsa, hash, sizeof(hash), tmp, sig_len);
    if(ret == 0) {
        printf("passed");
        u32Time = get_timer0_counter();
        printf("     takes %d.%d seconds,  %d ticks\n", u32Time / 1000000, u32Time / 1000,
g_u32Ticks);
    } else {
        printf("failed! ret[%d]\n", ret);
        mbedtls_ecdsa_free(&ecdsa);
        return ret;
    }

    mbedtls_ecdsa_free(&ecdsa);
    mbedtls_ecdsa_free(ecp);
    printf("\n mbedtls ECDSA done  \n");
```

3. 軟體與硬體需求

3.1 軟體需求

- BSP 版本
  - M4521_BSP_CMSIS_V3.00.003
- IDE 版本
  - Keil uVersion 6.10

3.2 硬體需求

- 電路元件
  - NuTiny-SDK-M4521S

4. 目錄資訊

EC_ M4521_ECC_Certification _V1.00	
Library	Sample code header and source files
CMSIS	Cortex® Microcontroller Software Interface Standard (CMSIS) by Arm® Corp.
Device	CMSIS compliant device header file
StdDriver	All peripheral driver header and source files
SampleCode	
ExampleCode	Source file of example code
ThirdParty	
mbedtls-3.1.0	A portable, easy to use, readable and flexible SSL library. Mbed TLS is available as open source under the Apache 2.0 license or the GPL 2.0 license. Its official website: https://tls.mbed.org/

圖 4-1 目錄資訊

5. 範例程式執行

1. 根據目錄資訊章節進入 ExampleCode 路徑中的 KEIL 資料夾，雙擊 *ECC_Certification.uvproj*。
2. 進入編譯模式介面
   - 編譯
   - 下載代碼至記憶體
   - 進入 / 離開除錯模式
3. 進入除錯模式介面
   - 執行代碼

6. 修訂紀錄

Date	Revision	Description
2022.08.01	1.00	初始發布。

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.

Please note that all data and specifications are subject to change without notice.
All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.