Merkle tree 代码说明

Merkle tree 是一种特殊的满二叉树，按照 RFC6962 标准来实现 merkle tree，对于 hash 函数的选用是 2048bit 安全性的 RSA 签名算法。

```
static uint8_t output_buf[2048 / 8];

uint8_t* _mbedtls_rsa_sign_test(const char* m)
{
    int ret;
    const char* msg = m;

    const char* pers = "rsa_sign_test";
    mbedtls_entropy_context entropy;
    mbedtls_ctr_drbg_context ctr_drbg;
    mbedtls_rsa_context ctx;

    /* 1. init structure */
    mbedtls_entropy_init(&entropy);
    mbedtls_ctr_drbg_init(&ctr_drbg);
    mbedtls_rsa_init(&ctx, MBEDTLS_RSA_PKCS_V21, MBEDTLS_MD_SHA256);

    /* 2. update seed with we own interface ported */
    printf("\n  . Seeding the random number generator...");

    ret = mbedtls_ctr_drbg_seed(&ctr_drbg, mbedtls_entropy_func, &entropy,
        (const unsigned char*)pers,
        strlen(pers));
    if (ret != 0) {
        printf(" failed\n  ! mbedtls_ctr_drbg_seed returned %d(-0x%04x)\n", ret, -ret);
        goto exit;
    }
    printf(" ok\n");

    /* 3. generate an RSA keypair */
    printf("\n  . Generate RSA keypair...");

    ret = mbedtls_rsa_gen_key(&ctx, mbedtls_ctr_drbg_random, &ctr_drbg, 2048, 65537);
    if (ret != 0) {
        printf(" failed\n  ! mbedtls_rsa_gen_key returned %d(-0x%04x)\n", ret, -ret);
        goto exit;
    }
    printf(" ok\n");
```

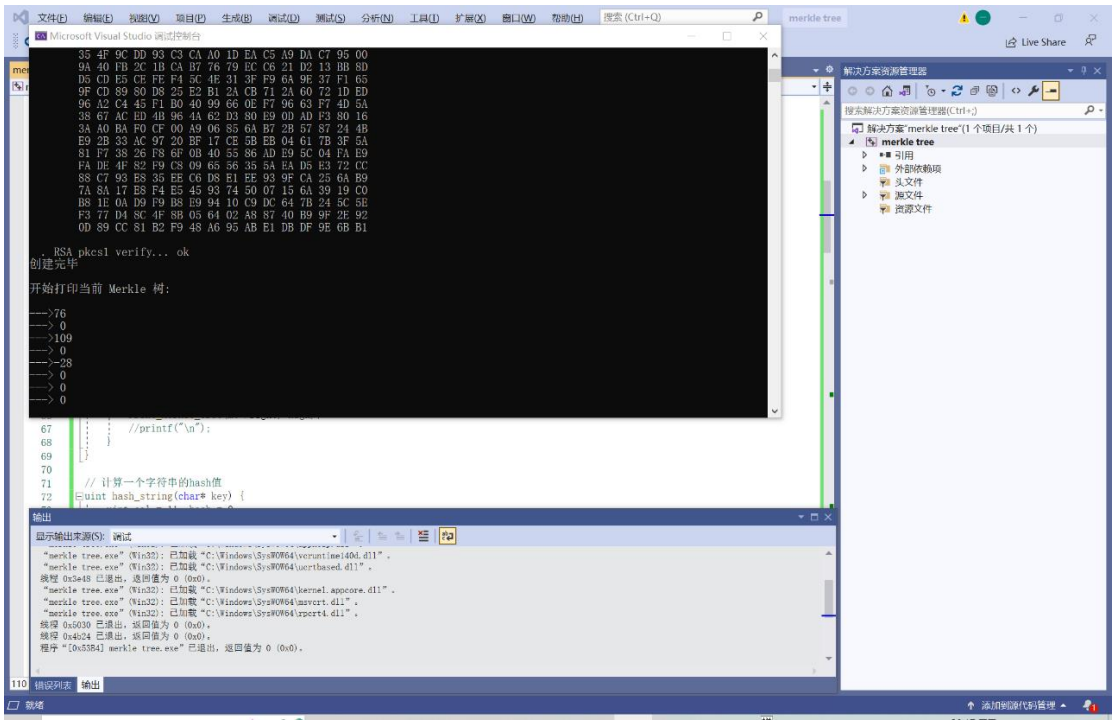
这些是 RSA 签名算法，使用了 mbedtls 库，具体的代码使用的是上学期密码学引论中使用的代码。

Print_MTree()是将树打印出来的函数，使用了递归的结构。

hash_Merkle()是用来计算每个节点 hash 值的函数。

Creat_MTree(MTNode* mt, char* arr, int nums, int tree_depth)是用来建立树的函数，mt 是根节点，arr 是用来建立树的数据，nums 是数据的数量，tree_deeth 是树高。同样采取了递归结构。

运行结果：


Microsoft Visual Studio 调试控制台
35 4F 9C DD 93 C3 CA A0 1D EA C5 A9 DA C7 95 00
9A 40 FB 2C 1B CA B7 76 79 EC C6 21 D2 13 BB 8D
D5 CD E5 CE FE F4 5C 4E 31 3F F9 6A 9E 37 F1 65
9F CD 89 80 D8 25 E2 B1 2A CB 71 2A 60 72 1D ED
96 A2 C4 45 F1 B0 40 99 66 0E F7 96 63 F7 4D 5A
38 67 AC ED 4B 96 4A 62 D3 80 E9 0D AD F3 B0 16
3A A0 BA F0 CF 00 A9 06 85 6A B7 2B 57 87 24 4B
E9 2B 33 AC 97 20 BF 17 CE 5B EB 04 61 7B 3F 5A
81 F7 38 26 F8 6F 0B 40 55 86 AD E9 5C 04 FA E9
FA DE 4F 82 F9 C8 09 65 56 35 5A EA D5 E3 72 CC
88 C7 93 E8 35 EE C6 D8 E1 EE 93 9F CA 25 6A B9
7A 8A 17 B8 F4 E5 45 93 74 50 07 15 6A 39 19 C0
B8 1E 0A D9 F9 B8 E9 94 10 C9 DC 64 7B 24 5C 5E
F3 77 D4 8C 4F 8B 05 64 02 A8 87 40 B9 9F 2E 92
0D 89 CC 81 B2 F9 48 A6 95 AB E1 DB DF 9E 6B B1
. RSA pkcs1 verify... ok
创建完毕
开始打印当前 Merkle 树:
--->76
---> 0
--->109
---> 0
--->-28
---> 0
---> 0
---> 0
//printf("\n");
// 计算一个字符串的hash值
uint hash_string(char* key) {
输出
显示输出来源(S): 调试
"merkle tree.exe" (Win32): 已加载 "C:\Windows\SysWOW64\ucrtbased.dll"。
线程 0x5e48 已退出，返回值为 0 (0x0)。
"merkle tree.exe" (Win32): 已加载 "C:\Windows\SysWOW64\kernel.appcore.dll"。
"merkle tree.exe" (Win32): 已加载 "C:\Windows\SysWOW64\msvcrt.dll"。
"merkle tree.exe" (Win32): 已加载 "C:\Windows\SysWOW64\rpcrt4.dll"。
线程 0x5030 已退出，返回值为 0 (0x0)。
线程 0x4b24 已退出，返回值为 0 (0x0)。
程序 "[0x5384] merkle tree.exe" 已退出，返回值为 0 (0x0)。
错误列表
输出
就绪
添加到源代码管理
解决方案资源管理器
解决方案"merkle tree"(1 个项目/共 1 个)
merkle tree
引用
外部依赖项
头文件
源文件
资源文件
Live Share