NXP Semiconductors
Application Notes

Document Number: AN12445
Rev. 4, 10/2020

非对称密码加速器CASPER

URL：https://www.nxp.com/docs/en/application-note/AN12445.pdf

目录

1. 简介

带有 RAM 共享 (CASPER) 外设的加密加速器和信令处理引擎可为非对称加密算法以及某些信号处理算法提供加速。

加速器速度更快，效率更高，功耗更低。它结合了速度以及使用更少的资源来执行大规模数学的艰巨任务。当加速器运行时，处理器可能处于空闲或休眠状态，或者它可能正在执行其他相关或不相关的任务。此外，系统可能能够使用较慢的时钟运行以降低功耗以提高能效。

本应用笔记介绍了 LPC5500 系列安全设备上的 CASPER：LPC55S6x、LPC55S2x、LPC55S1x 和 LPC55S0x。 由于这些设备使用完全相同的 CASPER，为简单起见，本文档中显示的示例基于 LPC55S69 的 SDK。

1.1. 非对称密码算法

此处定义的 CASPER 旨在成为一个非常通用的引擎，可以与软件结合应用于各种加密算法，包括非对称公钥（例如 RSA 和 ECC）以及相关的 Diffie-Hellman 密钥交换方法、生成器 指数和非标准大数算法。

1.2. 信号处理算法

CASPER 还可以选择性地参数化以执行信号处理操作，例如 FFT、DCT、iFFT、大多数矩阵运算以及基于 SIMD 的图形混合和缩放。

1.3. CASPER加速器模型及其提供的设施资源

加速器提供了六种设施资源来提高算法的效率/速度，通常是一个数量级的提升。 图 1 显示了 block 布局。

- AHB 总线和 Armv8-M 协处理器(CP) 接口，允许加载信息以执行操作。
- 快速共享内存访问，一次最多允许移动 128 位，如图 1 所示。
- 两个 32×32 乘法器。
- 二级加法器和寄存器组，允许进行 MAC 类型操作（乘后累加）。
- 一个掩码工具，允许通过侧信道计数计算而不需要将明文存储到触发器中。
- 根据操作的需要执行操作的状态机。

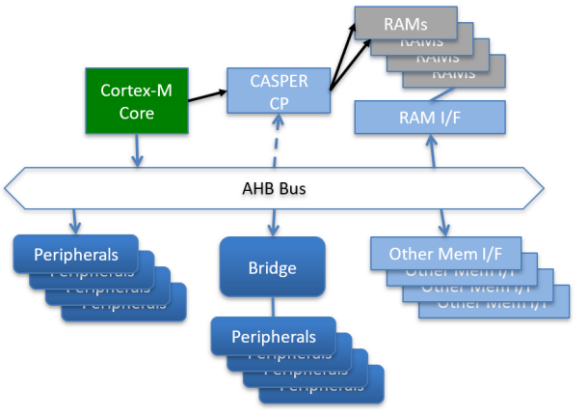


图 1. 显示 CASPER 如何融入特定系统

2. 方法

CASPER 加速器的方法基于图 2 中所示的基本属性。

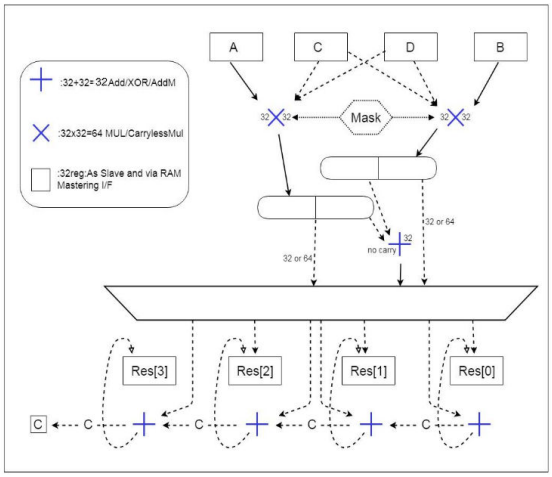


图 2. 显示 CASPER 加速器的框图

• 一组 4 个数据寄存器，每个 32 位(A/B/C/D)，用于馈送两个乘法器。乘法器可以将 XOR 掩码应用于侧信道用途。

• 一组 4 个结果寄存器(Res[3]/Res[2]/Res[1]/Res[0])，可与 4 个加法器一起使用，还可以执行 Add-Mask 和 XOR 运算。

• 可并行访问 2 或 4 个 RAM（最多 8 KB）。

— 该块对这些 RAM 使用了一个 RAM 接口，该接口也支持 AHB，以便应用程序可以随时访问 RAM。

— AHB 总线将 RAM 对视为通过交错组合（即一个是偶数字，一个是奇数字），而加速器将它们分开查看，允许一次性访问 64b 字对。

— 该块可以同时访问这两个或四个 bank，允许并行执行两个或四个操作——即一次 64 位或 128 位。

• 用于启动加速器的两个控制字（此处未显示）。

• 一个可选的掩码寄存器，用于创建一个 XOR 掩码以取消屏蔽 ABCD 和屏蔽输出以进行侧信道保护。

• 两个乘法器支持 32b × 32b，每个乘法器具有 64b 输出。

• 四个使用 ADD、XOR 的加法器。

• 执行全和时使用的进位位 (C)。

3. 操作

3.1. 模式

支持以下操作模式：

1. 64b×64b:(MUL)
2. 64b×64b + 64b×64b:(MUL+ADD)
3. 64b + 64b:(ADD)
4. 64b - 64b:(SUB)
5. 64b ^ 64b:(XOR)
6. 32b>>:(Shift Right)
7. 32b<<:(Shift Left)
8. Others: Copy, Remark, Fill, ZERO, Compare and so on

3.2. 两种示例模式示意执行和流程

3.2.1. **MUL (64b×64b = 128b)**

步骤：

1. Read ABCD
2. Step1:
    a. Compute DB and DA, sum=DBH+DAL
    b. Res[0] = DBL
    c. Res[1] = sum
    d. Res[2] = DAH
    e. Res[3] = 0
3. Step2:
    a. Compute CB and CA, sum=CBH+CAL
    b. Res[1] += CBL (generate Carry bit)
    c. Res[2] += sum + Carry bit (generate Carry bit)
    d. Res[3] = CAH + Carry bit
4. Step3:
    a. Write Res[3:0]

b. Done

步骤 1 可参考图 3。

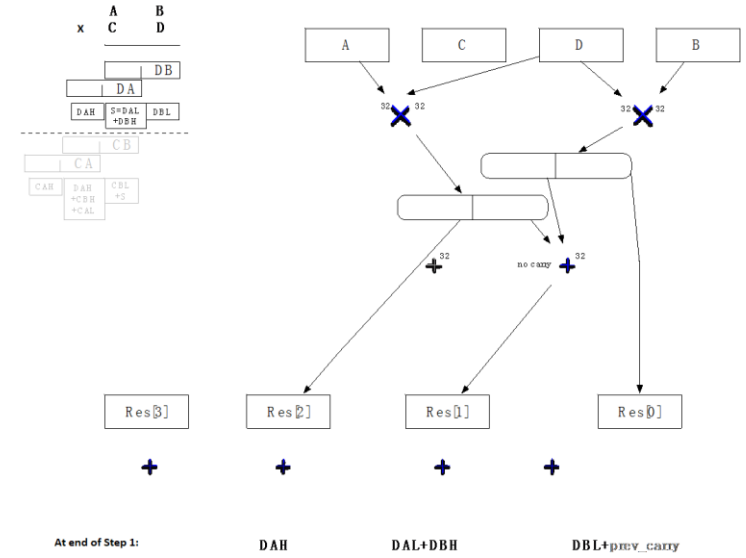


图 3. 64b×64b 的步骤 1 流程

步骤 2 可参考图 4。

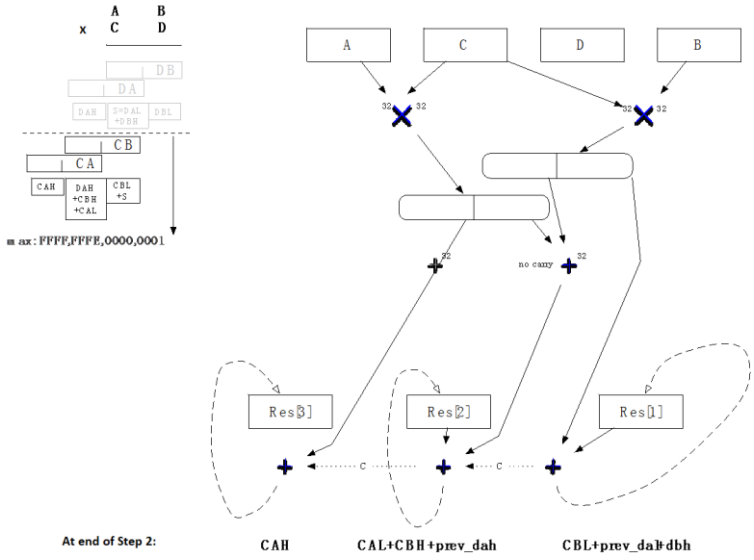


图 4. 64b×64b 的步骤 2 流程

3.2.2. **MUL+ADD (64b×64b + 64b×64b)**

步骤：

1. Read ABCD

2. Step1:
    a. Compute DB and DA, sum=DBH+DAL
    b. Res[0] += DBL (generate Carry bit)
    c. Res[1] += sum + Carry bit (generate Carry bit)
    d. Res[2] += DAH + Carry bit
    e. Res[3] += Carry bit
3. Step2:
    a. Compute CB and CA, sum=CBH+CAL
    b. Res[1] += CBL (generate Carry bit)
    c. Res[2] += sum + Carry bit (generate Carry bit)
    d. Res[3] += CAH + Carry bit
4. Step3:
    a. Write Res[3:0]
    b. Done

步骤 1 可参考图 5。

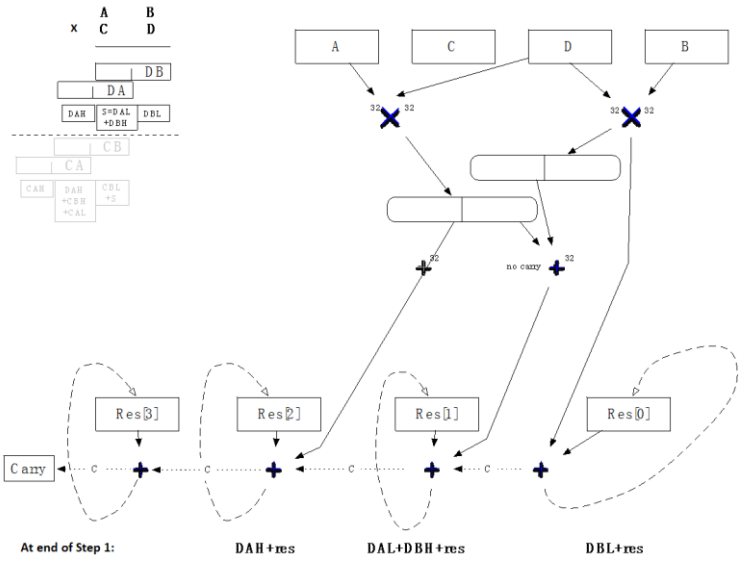
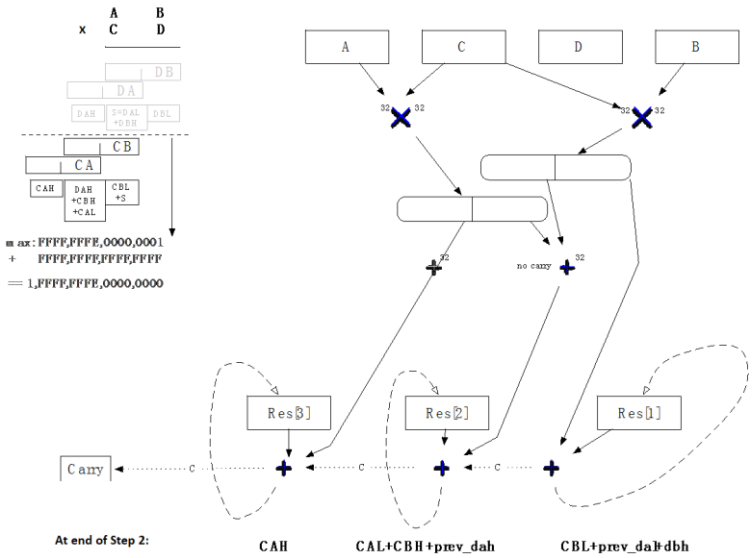


图 5. (64b×64b + 64b×64b) Step1 流程

步骤 2 可参考图 6。



4. RAM接口

RAM 模型设置为允许 2 个和 4 个 RAM（如下图 7 所示）。这意味着加速器可以同时访问 2 或 4 个 bank，允许对这些 RAM 进行 2 或 4 片并行访问，这意味着最多 128 位的读取、写入。但是 AHB 总线仍然是 32 位的访问。

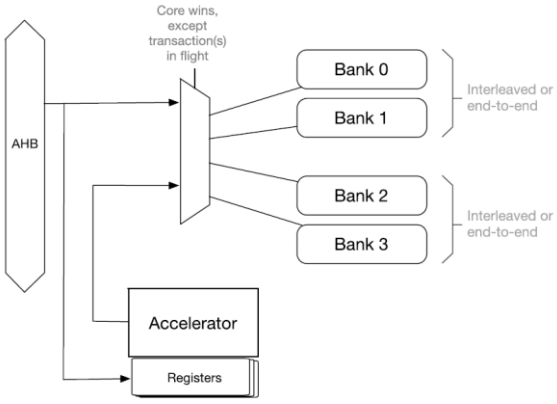


图 7. 带有加速器的系统中的 RAM

5. 性能数据

CASPER 加速器比用于加密目的的纯乘法器快几倍。各种用途的实际速度因算法、RAM 数量、是否交错以及软件如何放置其缓冲区而异。

LPC55S69 上 CASPER 加速和纯软件实现的性能如图 8 所示。

Operation	System clock:150MHz IDE: IAR8.32, Optimizations: high-speed-no size constraints	SW only		CASPER accelerated		Improvement(times)	
		RAM	Flash	RAM	Flash	RAM	Flash
Signing	ECDSA-secp256r1(ms/sign)	136.43	333.33	76.92	142.86	1.77	2.33
Verification	ECDSA-secp256r1(ms/verify)	250.00	598.80	81.10	149.93	3.08	3.99
Key exchange	ECDHE-secp256r1(ms/handshake)	250.00	500.00	136.43	250.00	1.83	2.00
Key exchange	ECDH-secp256r1(ms/handshake)	136.43	300.30	71.43	130.38	1.91	2.30
Signing	RSA-1024(ms/private)	130.38	250.00	130.38	272.48	-	-
Verification	RSA-1024(ms/public)	4.24	8.90	1.31	1.81	3.24	4.93
Signing	RSA-2048(ms/private)	598.80	1000.00	598.80	1000.00	-	-
Verification	RSA-2048(ms/public)	15.54	31.92	4.14	5.03	3.76	6.35

图 8. 由软件和 CASPER 实现的非对称加密算法的性能比较

评价：

1. CASPER 性能与 LPC5500 系列其他安全器件如 LPC55S2x、LPC55S1x、LPC55S0x 等相似

2.IDE 版本：IAR8.32.2

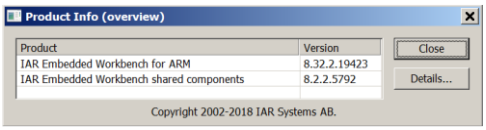


图 9. IAR 版本

3. 优化器：高频、速度快、无尺寸限制

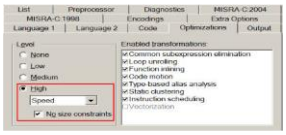


图 10. IAR 优化器

4. SDK 版本：2.6.3 (2019-10-11)

SDK 标签：REL_SDK_NIOBE4_2.6.3_RFP3.0_RC2_3

5. 系统时钟：150 MHz

6. 示例版本：项目的发布版本。

7. - 代表没有 CASPER 加速，硬件和软件效率是一样的。

8.如果在 flash 中运行，需要安装 LPC55S69_cm33_core0_flash.icf。

如果在 RAM 中运行，则需要安装 LPC55S69_cm33_core0_ram.icf。

9. 开发板：LPCXpresso55S69、LPC55S69-EVK

10.硅版本：1B

6. SDK implementations

CASPER 的例子可以在 SDK 包中找到。

从恩智浦网站下载最新版本的 SDK 后，在以下位置打开 CASPER 示例：

SDK_2.6.3_LPCXpresso55S69\boards\lpcxpresso55s69\driver_examples\casper\cm33_core0

编译项目并打开 casper.c 文件。有一些应用程序功能可以分为两个应用程序：

- ModExp 算法
- 椭圆曲线 Secp256r1 乘法

6.1. ModExp 算法

模幂运算是一种算法，其中在模块上执行幂运算。 它在计算机科学中很有用，特别是在公钥密码学领域。

以下示例说明如何使用公钥（包括 E 和 N）验证签名，如图 11 中的公式。

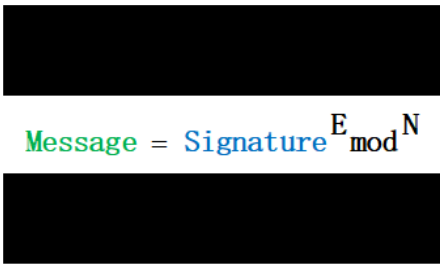


图 11. 验证签名公式

函数代码示例如图 12 所示。

```
/* ModExp test */
CASPER_ModExp(CASPER, (void *)signature0, (void *)pubkey0, sizeof(plaintext) / sizeof(uint32_t), pub_e, plaintext);
TEST_ASSERT(memcmp(plaintext0, plaintext, sizeof(plaintext)) == 0);
PRINTF("ModExp Test pass.\r\n");
```

图 12. ModEXP 代码

实现过程包括一系列复杂的数据转换。它基于经典的 ModExp 算法，包括 Montgomery 模乘法等。详情可以上网查查。最后，该算法使用基本的乘法、加法和减法算法。这些算法可以通过 CASPER 来实现。一些基本的应用程序代码如图 13 所示。

```
Accel_SetABCD_Addr(CA_MK_OFF(&v[0]), CA_MK_OFF(u));
Accel_crypto_mul(Accel_IterOpcodeResaddr(N_wordlen / 2 - 1, kCASPER_OpMul6464NoSum, CA_MK_OFF(&w_out[0])));
Accel_done();

Accel_SetABCD_Addr(CA_MK_OFF(&v[j]), CA_MK_OFF(u));
Accel_crypto_mul(Accel_IterOpcodeResaddr(N_wordlen / 2 - 1, kCASPER_OpMul6464Sum, CA_MK_OFF(&w_out[j])));
Accel_done();

Accel_SetABCD_Addr(CA_MK_OFF(Nmod), 0);
Accel_crypto_mul(Accel_IterOpcodeResaddr(N_dwordlen - 1, kCASPER_OpSub64, CA_MK_OFF(Rp)));
Accel_done();

carry = GET_DWORD(&w64[N_dwordlen]);
Accel_SetABCD_Addr(CA_MK_OFF(&b64[i]), CA_MK_OFF(a64));
Accel_crypto_mul(Accel_IterOpcodeResaddr(N_dwordlen - 1, kCASPER_OpMul6464FullSum, CA_MK_OFF(w64)));
Accel_done();
```

图 13. CASPER 应用

由于 CASPER 的加速器功能，RSA 签名验证会很快。在函数中，有一些 CASPER 操作。如图 14 所示，这些操作对应于操作中描述的操作模式。

```
typedef enum _casper_operation
{
    kCASPER_OpMul6464NoSum = 0x01, /*! Walking 1 or more of J loop, doing r=a*b using 64x64=128*/
    kCASPER_OpMul6464Sum =
        0x02, /*! Walking 1 or more of J loop, doing c,r=r+a*b using 64x64=128, but assume inner j loop*/
    kCASPER_OpMul6464FullSum =
        0x03, /*! Walking 1 or more of J loop, doing c,r=r+a*b using 64x64=128, but sum all of w. */
    kCASPER_OpMul6464Reduce =
        0x04,                 /*! Walking 1 or more of J loop, doing c,r[-1]=r+a*b using 64x64=128, but skip 1st write*/
    kCASPER_OpAdd64 = 0x08, /*! Walking add with off_AB, and in/out off_RES doing c,r=r+a+c using 64+64=65*/
    kCASPER_OpSub64 = 0x09, /*! Walking subtract with off_AB, and in/out off_RES doing r=r-a uding 64-64=64, with last
                               borrow implicit if any*/
    kCASPER_OpDouble64 = 0x0A, /*! Walking add to self with off_RES doing c,r=r+r+c using 64+64=65*/
    kCASPER_OpXor64 = 0x0B,    /*! Walking XOR with off_AB, and in/out off_RES doing r=r^a using 64^64=64*/
    kCASPER_OpShiftLeft32 =
        0x10, /*! Walking shift left doing r1,r=(b*D)|r1, where D is 2^amt and is loaded by app (off_CD not used)*/
    kCASPER_OpShiftRight32 = 0x11, /*! Walking shift right doing r,r1=(b*D)|r1, where D is 2^(32-amt) and is loaded by
                                      app (off_CD not used) and off_RES starts at MSW*/
    kCASPER_OpCopy = 0x14,         /*! Copy from ABoff to resoff, 64b at a time*/
    kCASPER_OpRemask = 0x15,       /*! Copy and mask from ABoff to resoff, 64b at a time*/
    kCASPER_OpCompare = 0x16,      /*! Compare two arrays, running all the way to the end*/
    kCASPER_OpCompareFast = 0x17,  /*! Compare two arrays, stopping on 1st !=*/
} casper_operation_t;
```

图 14. CASPER 操作

6.2. 椭圆曲线乘法

这些函数执行 ECC secp384r1 点单标量乘法[resX; resY] = 标量_[X; Y]和 ECC secp384r1 点双标量乘法[resX; resY] = 标量 1 * [X1; Y1] + 标量 2 * [X2; Y2]。它们是椭圆曲线密码术(ECC)的基础。关于 ECC 的任何细节，你可以在网上进行研究。

功能代码如图 15 所示。

```
CASPER_ECC_SECP384R1_MulAdd(CASPER, c3, c4, &test_ecddoublemul_base[0][0],
                            &test_ecddoublemul_base[0][NUM_LIMBS], &test_ecddoublemul_scalars[i][0],
                            &test_ecddoublemul_base[1][0], &test_ecddoublemul_base[1][NUM_LIMBS],
                            &test_ecddoublemul_scalars[i][NUM_LIMBS]);
CASPER_ECC_SECP384R1_Mul(CASPER, X1, Y1, &test_ecmulans[0][0], &test_ecmulans[0][12], test_ecmulscalar[i]);
```

图 15. 椭圆曲线乘法

与 ModExp 一样，ECC 乘法的基本实现基于 CASPER API，如图 16 所示。

```
/*  we are reducing, so the 1st [0th] 64 bit value product is tossed, but we */
/*  need its carry. We let the accel do this separately - really need a mode to */
/*  do this "reduce" since it is natural */
carry = GET_DWORD(&w64[N_dwordlen]);
Accel_SetABCD_Addr(CA_MK_OFF(m64), CA_MK_OFF(&N64[0]));
Accel_crypto_mul(Accel_IterOpcodeResaddr(N_dwordlen - 1, kCASPER_OpMul6464FullSum, CA_MK_OFF(&w64[0])));
Accel_done();
carry = (GET_DWORD(&w64[N_dwordlen]) < carry);

Accel_SetABCD_Addr(CA_MK_OFF(&w64[1]), 0);
Accel_crypto_mul(Accel_IterOpcodeResaddr(N_dwordlen - 1, kCASPER_OpCopy, CA_MK_OFF(&w64[0])));

Accel_done();
SET_DWORD(&w64[N_dwordlen], (GET_DWORD(&w64[N_dwordlen + 1]) + carry));
```

图 16. CASPER API

最后，运行示例代码后，即可在 CommAssistant 上获取打印字符串，如图 17。

```
ModExp Test pass.
Casper ECC Demo P256

Round: 0
Round: 1
Round: 2
Round: 3
Round: 4
Round: 5
Round: 6
Round: 7
All EC scalar multiplication tests were succesfull.
Round: 0
Round: 1
Round: 2
Round: 3
Round: 4
Round: 5
Round: 6
Round: 7
All EC double scalar multiplication tests were succesfull.
```

图 17. CASPER 示例结果

7. CASPER在mbedTLS中的使用

mbedTLS 的例子可以在 SDK 包中找到，其位置如下：

SDK_2.6.3_LPCXpresso55S69\boards\lpcxpresso55s69\mbedtls_examples\mbedtls_benchmark\cm33_core0

演示应用程序执行包括对称和非对称加密在内的加密算法。 CASPER HW 在 RSA-1024 加密、ECDSA-secp256r1 签名和验证、ECDHE-secp256r1 密钥交换、ECDH-secp256r1 密钥交换中加速。

下载并运行代码后，调试端口如图 18 所示。

```
mbedTLS version 2.13.1
fsys=150000000
Using following implementations:
  SHA: HASHCRYPT HW accelerated
  AES: HASHCRYPT HW accelerated
  AES GCM: Software implementation
  DES: Software implementation
  Asymmetric encryption: CASPER HW accelerated


  MD5                       :  3772.60 KB/s,   38.52 cycles/byte
  SHA-1                     :  34736.80 KB/s,    3.93 cycles/byte
  SHA-256                   :  36477.19 KB/s,    3.72 cycles/byte
  SHA-512                   :  344.32 KB/s,  426.39 cycles/byte
  3DES                      :   94.61 KB/s,  1564.47 cycles/byte
  DES                       :  261.97 KB/s,  561.00 cycles/byte
  AES-CBC-128               :  30945.76 KB/s,    4.47 cycles/byte
  AES-CBC-192               :  30742.89 KB/s,    4.50 cycles/byte
  AES-CBC-256               :  30586.30 KB/s,    4.52 cycles/byte
  AES-GCM-128               :  424.11 KB/s,  345.92 cycles/byte
  AES-GCM-192               :  420.78 KB/s,  348.66 cycles/byte
  AES-GCM-256               :  419.69 KB/s,  349.54 cycles/byte
  AES-CCM-128               :  725.79 KB/s,  201.80 cycles/byte
  AES-CCM-192               :  706.72 KB/s,  207.26 cycles/byte
  AES-CCM-256               :  700.73 KB/s,  209.04 cycles/byte
  CTR_DRBG (NOPR)           :  2266.46 KB/s,   64.36 cycles/byte
  CTR_DRBG (PR)             :  1579.29 KB/s,   92.53 cycles/byte
  HMAC_DRBG SHA-1 (NOPR)    :  543.86 KB/s,  269.55 cycles/byte
  HMAC_DRBG SHA-1 (PR)      :  498.77 KB/s,  293.98 cycles/byte
  HMAC_DRBG SHA-256 (NOPR)  :  840.45 KB/s,  174.20 cycles/byte
  HMAC_DRBG SHA-256 (PR)    :  729.89 KB/s,  200.65 cycles/byte
  RSA-1024                  :  196.33  public/s
  RSA-1024                  :    2.33 private/s
  DHE-2048                  :    0.11 handshake/s
  DH-2048                   :    0.11 handshake/s
  ECDSA-secp256r1           :    3.33 sign/s
  ECDSA-secp256r1           :    3.33 verify/s
  ECDHE-secp256r1           :    2.00 handshake/s
  ECDH-secp256r1            :    3.67 handshake/s
```

图 18. CASPER HW 加速结果

如果在 LPC55S69_cm33_core0_features.h 中将 FSL_FEATURE_SOC_CASPER_COUNT 设置为 0，它将更改为软件实现。结果如图 19 所示。

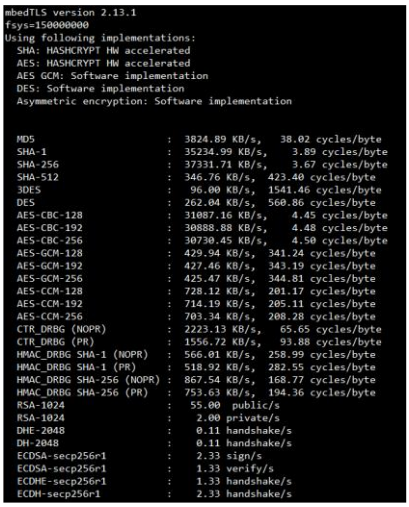


图 19. 软件实现结果

如果在 LPC55S69_cm33_core0_features.h 中将 FSL_FEATURE_SOC_CASPER_COUNT 设置为 1 并逐步调试，可以找到如下函数调用：

1. RSA-1024 的实现：196.33 public/s。

CASPER_ModExp() 是 CASPER 低驱动 API，用于 RSA-1024 加密，如图 20 中的 Call Stack 所示。

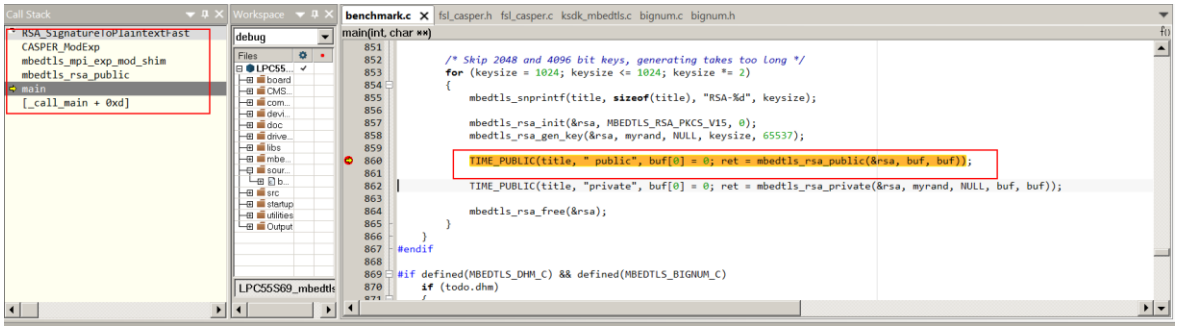


图 20. RSA-1024 加密实现

2. ECDSA-secp256r1 的实现：3.33 sign/s。

CASPER_ECC_SECP256R1_Mul()是 CASPER 低层驱动 API，用于 ECDSA-secp256r1 签名，如图 21 的 Call Stack 所示。

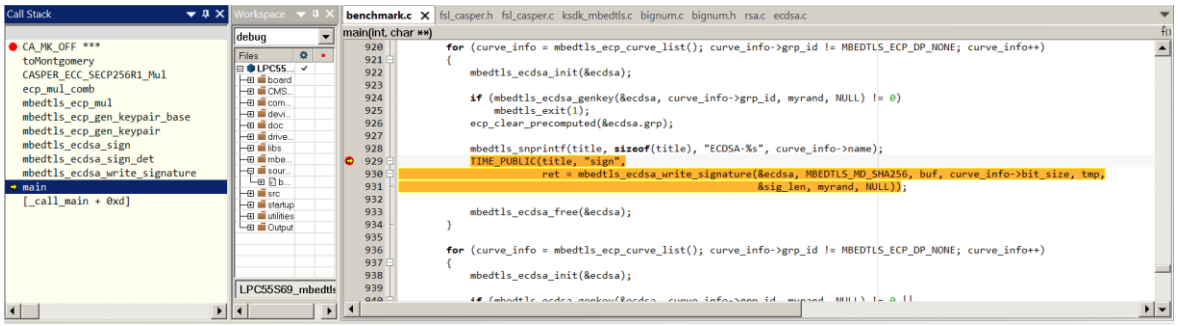


图 21. ECDSA-secp256r1 签名实现

3. ECDSA-secp256r1 的实现：3.33 verify/s。

CASPER_ECC_SECP256R1_MulAdd()是 CASPER 低层驱动 API，用于 ECDSA-secp256r1 验证，如图 22 Call Stack 所示。

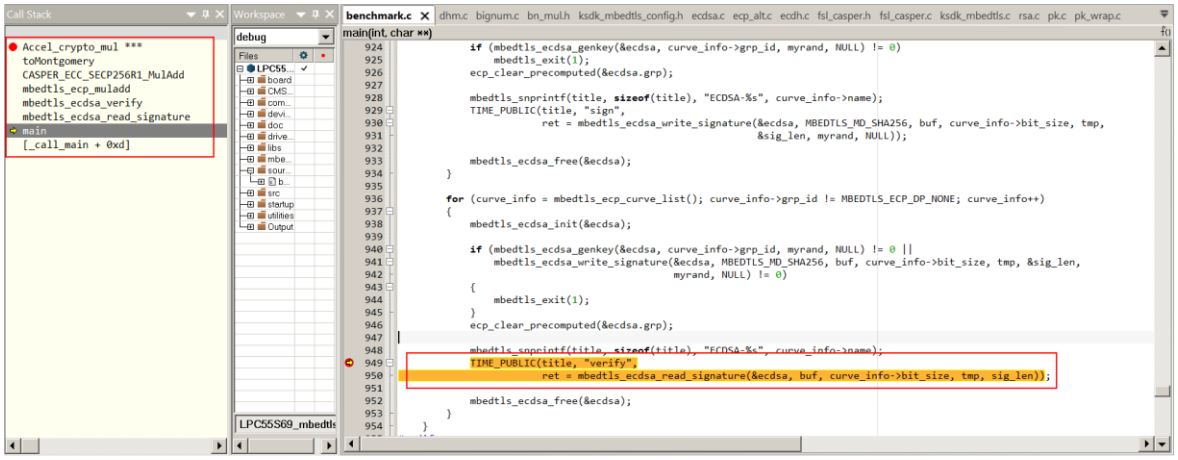


图 22. ECDSA-secp256r1 验证实现

4. ECDHE-secp256r1 的实现：2.00 handshake/s

CASPER_ECC_SECP256R1_Mul()是 CASPER 低层驱动 API，用于 ECDHE-secp256r1 密钥交换，如图 23 的 Call Stack 所示。

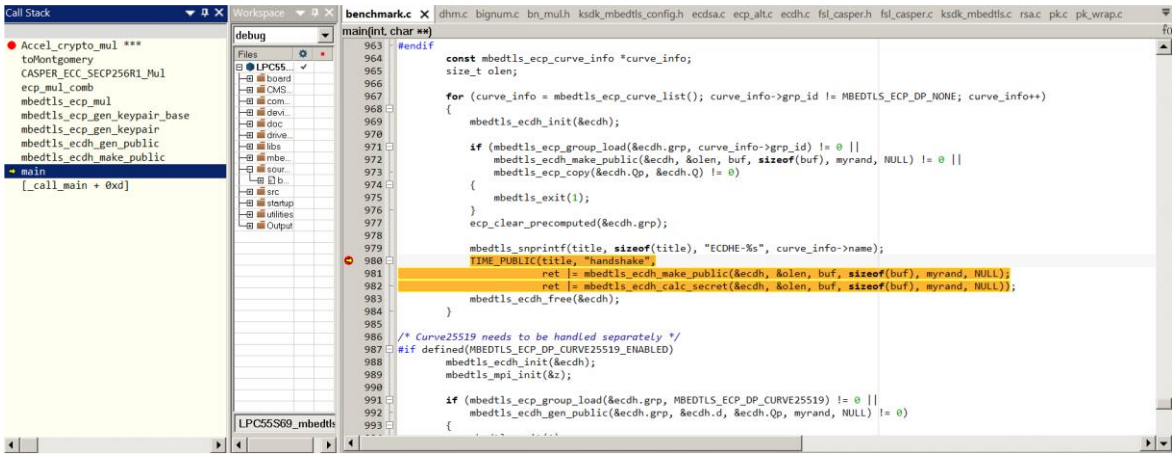


图 23. ECDHE-secp256r1 密钥交换实现

5. ECDH-secp256r1 的实现：3.67 handshake/s。

CASPER_ECC_SECP256R1_Mul() 是 CASPER 低驱动程序 API，用于 ECDH-secp256r1 密钥交换，如图 24 中的 Call Stack 所示。

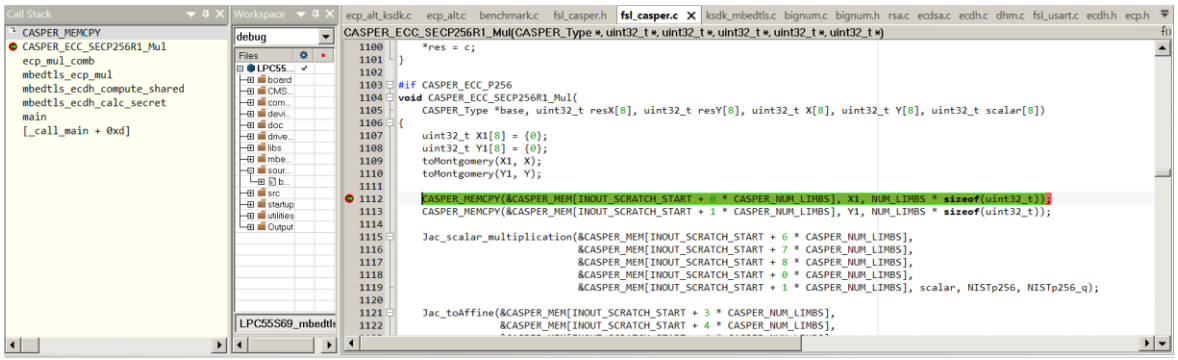


图 24. ECDH-secp256r1 密钥交换实现

8. 修订历史

表 1. 修订历史

版本	日期	描述
0	2019 年 4 月 20 日	首次发布
1	2019 年 7 月 15 日	一般更新
2	2019 年 10 月 23 日	参数更新
3	2020 年 1 月 7 日	包括 LPC55S2x/1x
4	2020 年 10 月 27 日	添加了 LPC55S0x

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 10/2020
Document identifier: AN12445