产品名称Product name	密级Confidentiality level
加密卡	
产品版本Product version	Total 24 pages共24页
V1.1	

测试报告

(仅供内部使用)

拟制:	马亚飞	日期：	2018.4.25
审核:		日期：	
审核:		日期：	
批准:		日期：	

深圳数字电视国家工程实验室股份有限公司

修订记录

日期	修订版本	描述	作者
2018年4月20日	1.0	创建	马亚飞
2018年4月25日	1.1	完善测试报告	马亚飞

1 目录

1. 目的

- 统计加密卡的功能测试结果。
- 统计加密卡的性能测试结果。
- 统计疲劳测试的结果。

2. 测试范围

功能测试不针对所有加密卡进行，随机抽取 10 张加密卡进行测试。性能测试结果也随机抽取 4 张进行,分别测试单卡和双卡的性能指标。

疲劳测试则覆盖所有生产的加密卡。

3. 测试向量

在 openssl1.0.2g 版本中不支持异步 API 调用，如需增加异步使用方式，需更新 openssl 的版本为 openssl1.1.0e。因此本文测试向量主要包括同步的 HASH、AES、RSA、ECDSA 算法，具体见下表。

注：目前有两种驱动可选，版本 1 不支持多卡，但支持 HASH、AES 的硬件加速；版本 2 支持多卡，但 HASH、AES 采用的是主芯片的对称加密引擎实现，本文所列的数据均基于版本 2 进行测试的结果。

算法	密钥长度	接口	模式
RSA	1024	公钥加密	RSA_PKCS1_PADDING
			RSA_PKCS1_OAEP_PADDING
		私钥解密	RSA_PKCS1_PADDING
			RSA_PKCS1_OAEP_PADDING
		签名	sha1WithRSA
			md5_sha1
		验证签名	sha1WithRSA
			md5_sha1
	2048	公钥加密	RSA_PKCS1_PADDING
			RSA_PKCS1_OAEP_PADDING
		私钥解密	RSA_PKCS1_PADDING
			RSA_PKCS1_OAEP_PADDING
		签名	sha1WithRSA
			md5_sha1
		验证签名	sha1WithRSA
			md5_sha1
	4096	公钥加密	RSA_PKCS1_PADDING
			RSA_PKCS1_OAEP_PADDING
		私钥解密	RSA_PKCS1_PADDING

			RSA_PKCS1_OAEP_PADDING
		签名	sha1WithRSA
			md5_sha1
		验证签名	sha1WithRSA
			md5_sha1

算法	密钥长度	接口	模式
SHA256	/	/	空输入
			正常输入
SHA1	/	/	空输入
			正常输入

算法	密钥长度	接口	模式
aes	128	加密	cbc
			ecb
		解密	cbc
			ecb

算法	密钥长度	接口	模式
ECDSA	256	签名	secp256k1
			secp256r1/X9_62_prime256v1
		验签	secp256k1
			secp256r1/X9_62_prime256v1

4. 测试环境

4.1. 硬件环境

性能测试主机配置采用两种配置进行对比测试，其配置如下表所示：

配置	主机一	主机二
CPU	Intel(R) Pentium(R) G840 (2.8GHz，双核，未开启超线程)	Intel(R) Core(TM) i7-7700 （3.6GHz，超线程开启 8 核）
内存	4G，DDR3	8G,DDR3
硬盘	DMA 模式开启	DMA 模式开启
卡槽 1	PCIe X16	PCIe X16
卡槽 2	PCIe X1 EdgeFree	PCIe X16

4.2. 软件环境

1. Linux version 4.4.0-62-generic (buildd@lcy01-30) (gcc version 5.4.0 20160609 (Ubuntu 5.4.0-6ubuntu1~16.04.4)) ;
2. cryptodev 编译为两种引用方式：builtin 和动态引擎。功能和性能测试采用动态引擎加载机制进行测试，疲劳测试则采用的是 builtin 的方式进行的测试。
3. 主机安装 openssl，版本为 1.0.2g;

5. 测试流程

5.1. 功能测试流程

使用 openssl 命令进行测试，编译支持动态引擎的 openssl_1.0.2g 版本的 openssl 可执行程序，然后编译 cryptodev 引擎动态库，通过命令行的方式来进行加载切换。测试输入文件包括两个文件，分别表示正常输入（plain.txt）和空输入(null.txt)。分别执行不加载引擎(CPU)和加载引擎(加密卡)的相关命令，并比较二者的结果是否一致。例如 sha1 算法，执行的命令和结果如下所示：

```
$ ./openssl sha1 -engine ./libeng_cryptodev.so plain.txt
engine "cryptodev" set.
SHA1(plain.txt)= 7317399078361350034a7b769c6ea479ca1ebc4a
$ ./openssl sha1 plain.txt
SHA1(plain.txt)= 7317399078361350034a7b769c6ea479ca1ebc4a
```

可以看出二者得到的值一致，表示 sha1 算法功能验证通过。其他 hash 算法也采用相同操作进行验证。对 aes 算法则采用“软加硬解”和“硬加软解”两种方式同时验证，即用 CPU 加密，用加密卡解密；用加密卡加密，用 CPU 解密。功能测试中用到的命令如下所示：

```
$ ./openssl md5/sha1/sha256 xxxxxxx
$ ./openssl dgst -hmac xxxx -sha1/sha256 xxxxxx
$ ./openssl aes-128-cbc/ecb/xx -e -base64 -in xxxx -iv xxxx -k xxxx
$ ./openssl aes-128-cbc/ecb/xx -d -base64 -in xxxx -iv xxxx -k xxxx
$ ./openssl rsautl -encrypt -pkcs/oaep -inkey xxx -in xxx  -pubin
$ ./openssl rsautl -decrypt -pkcs/oaep -inkey xxx -in xxx
$ ./openssl dgst -sign xxx -sha256 -out xxx
```

```
$ ./openssl dgst -verify xxxx -sha256 -signature xxxx
```

5.2. 性能测试流程

1 性能测试采用 openssl 自带的 speed 测速模块，来统计各个算法运行的速度；

2 加上并行运算的参数，统计其性能的变化。

```
$ openssl speed rsa/ecdsap256 -elapsed -multi xx  ---CPU 测试
$ ./openssl speed rsa/ecdsap256 -elapsed  -multi xx  ---加密卡测试
```

5.3. 疲劳测试流程

疲劳测试根据测试向量编写，设计支持多线程机制，通过参数来设定测试范围。具体用法如下所示：

```
usage: newtest [args ...]

 -digest      - digest test vector: sha1  sha256  hmac-sha1  hmac-sha256

 -enc         - enc test vector:    aes-128-cbc  aes-128-ecb  aes-192-cbc  aes-192-ecb  aes-256-cbc
                                    aes-256-ecb

 -ecdsa       - ecdsa test vector:  secp256k1  secp256r1

 -rsa         - rsa opertion, key size: 1024, 2048, 4096
 -all         - test all the fixed vectors
 -mtparams    - will test valid or invalid inputs
 -threads arg - number of threads, max is 200
 -loops arg   - number of operations, per thread
 -engine arg  - 1: software, 2: hardware, default is hardware
 -help        - show this message
```

其中-digest 参数会执行 hash 算法，包括 sha1，sha256，hmac-sha1，hmac-sha256，循环并依次执行这几个摘要算法。

-enc 则执行 aes 加解密操作，支持 6 种模式，循环并依次执行加密、解密操作，并验证操作结果是否正确。

-ecdsa 参数则执行 ecdsa 签名、验签步骤，分别执行 ecdsap256k1 和 ecdsap256r1 两条曲线进行测试，判断结果是否正确。密钥采用预置在代码中的方式进行，不临时生成。

-rsa 参数执行 rsa 加密、解密、签名、验签四个过程，依次包括 rsa1024，rsa2048，rsa4096。密钥采用预置在代码中的方式进行，不临时生成。

-all 参数则会执行全部的测试选项。

-threads 用来设定并发线程数，最大支持 200。

-loops 参数来设置每个线程中操作执行的循环次数，该操作是指 enc、ecdsa、rsa、all 设定的步骤。

-engine 可以用来设定测试程序运行时的引擎，默认采用 cryptodev 引擎。

测试执行时，测试程序会根据参数执行操作，并通过参数来设定执行压力，通常设置为 150

个线程，每个线程循环次数为 1000 次，测试程序会不断执行设定的操作。

注意：测试输出中，速度的统计不具备性能说明意义，仅作为上下文判断是否速度有明显差异。

6. 测试结果

6.1. 功能测试

执行功能测试脚本，看是否有错误出现，通过测试，发现所有测试向量均正确，无错误出现。

表 1 功能测试结果

Type		Result
hash	sha1	√
	sha256	√
	md5	√
	hmac-sha1	√
	hmac-sha256	√
AES	aes-128-cbc	√
	aes-128-ecb	√
	aes-192-cbc	√
	aes-192-ecb	√
	aes-256-cbc	√
	aes-256-ecb	√
RSA (1024, 2048, 4096)	encrypt	√
	decrypt	√
	sign	√

	verify	√
ECDSA	sign	√
(secp256)	verify	√

6.2. 基于主机一测试结果（性能测试）

6.2.1. 单进程测试

摘要和对称 aes 算法性能测试，使用 openssl speed 命令进行统计，测试命令如下和结果如下所示，下表中的数字表示每秒处理的 1000K 字节数，单位为 MB/s

```
$ ./openssl speed -elapsed md5 sha1 sha256 sha512 aes-128-cbc ###加密卡计算
$ openssl speed -elapsed md5 sha1 sha256 sha512 aes-128-cbc    #### CPU 计算
```

表 1 CPU 计算结果(MB/s)

类型	16bytes	64bytes	256bytes	1024bytes	8192bytes
sha1	53.377	144.847	300.222	406.971	453.872
sha256	40.061	88.108	151.280	183.751	195.974
sha512	26.794	107.171	159.943	221.954	250.301
md5	48.511	143.525	313.362	445.961	508.644
aes-128-cbc	85.576	92.317	93.934	94.621	94.784

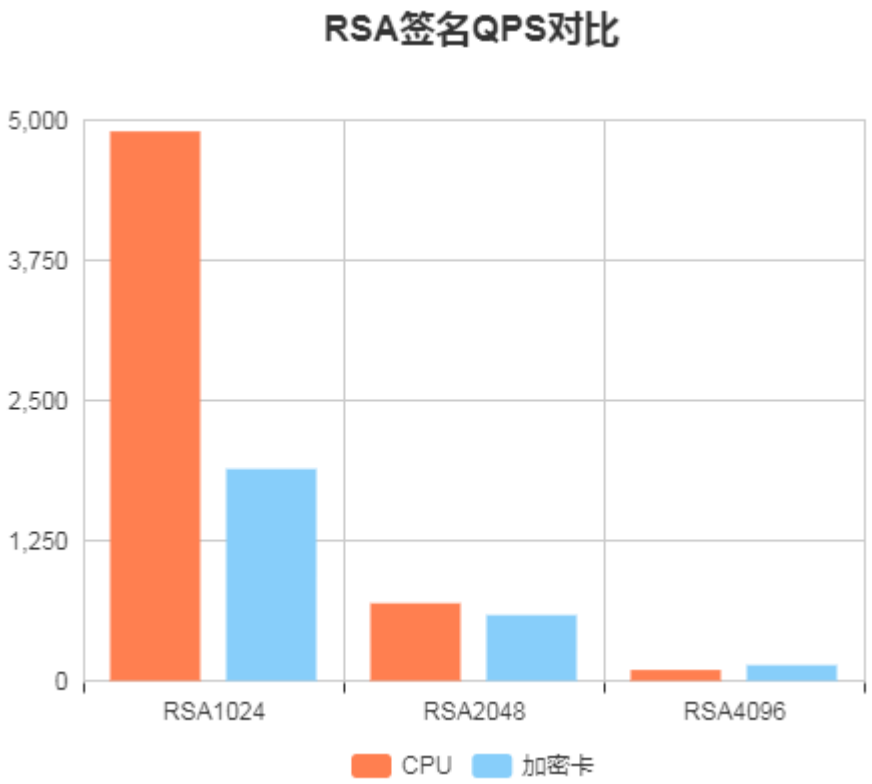
表 2 加密卡计算结果

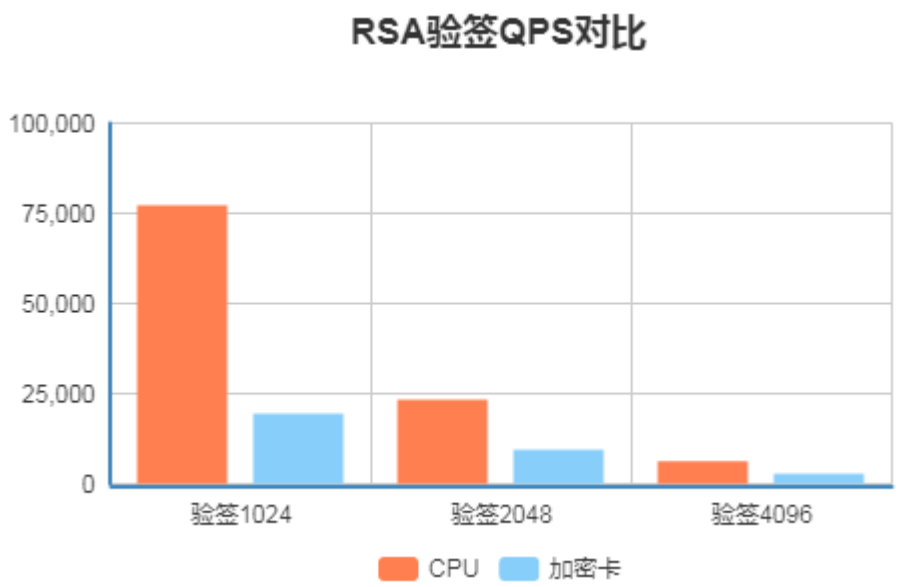
类型	16bytes	64bytes	256bytes	1024bytes	8192bytes
sha1	7.406	27.119	88.087	200.835	312.907
sha256	41.366	89.208	152.022	184.481	196.141
sha512	27.531	110.987	160.985	222.674	250.423
md5	7.935	29.760	100.589	243.607	411.475

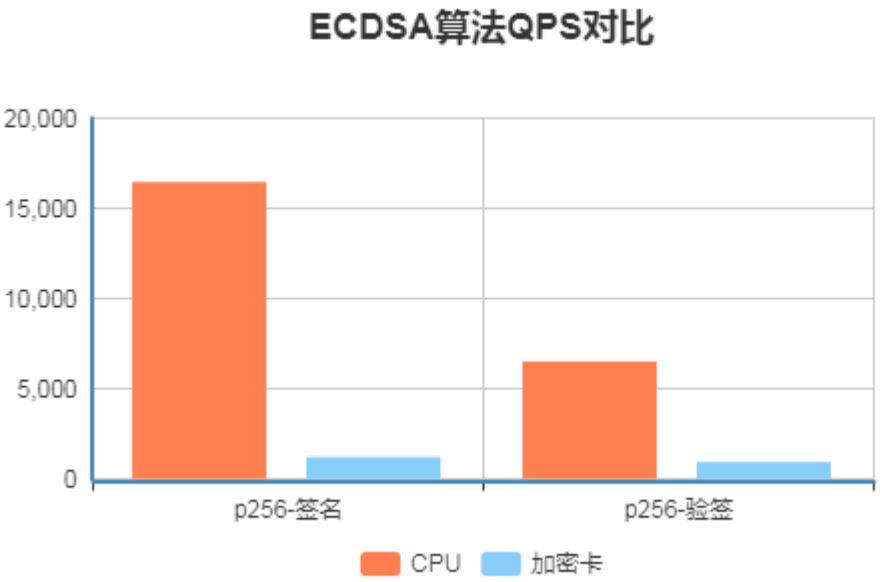
aes-128-cbc	86.006	92.414	93.995	94.652	94.827

通过上面 2 张表格的对比，可以发现这几种算法的性能在使用 CPU 和加密卡的时候基本没有变化。

QPS 表示每秒请求次数，即 speed 命令执行结果中的速度统计，下面几幅图为基于主机一测试的结果，测试为单卡、单进程测试条件对比，包括 rsa（1024，2048，4096）和 ecdsa（p256）算法。







通过上面两幅图可以看出在单进程测试条件下，加密卡的性能并没有发挥出来，不管是 RSA 还是 ECC 的算法都要比软件执行速度要慢。

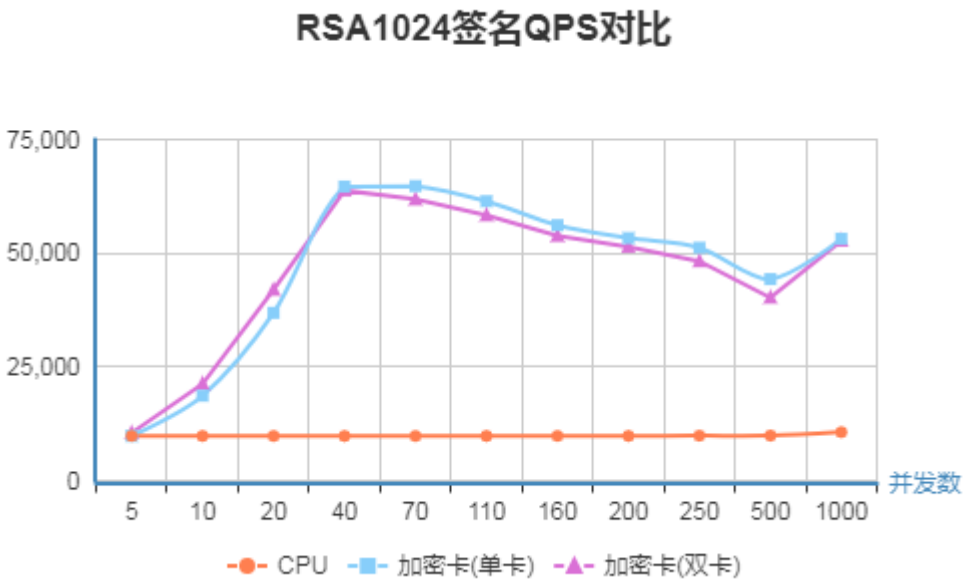
6.2.2. 多进程测试

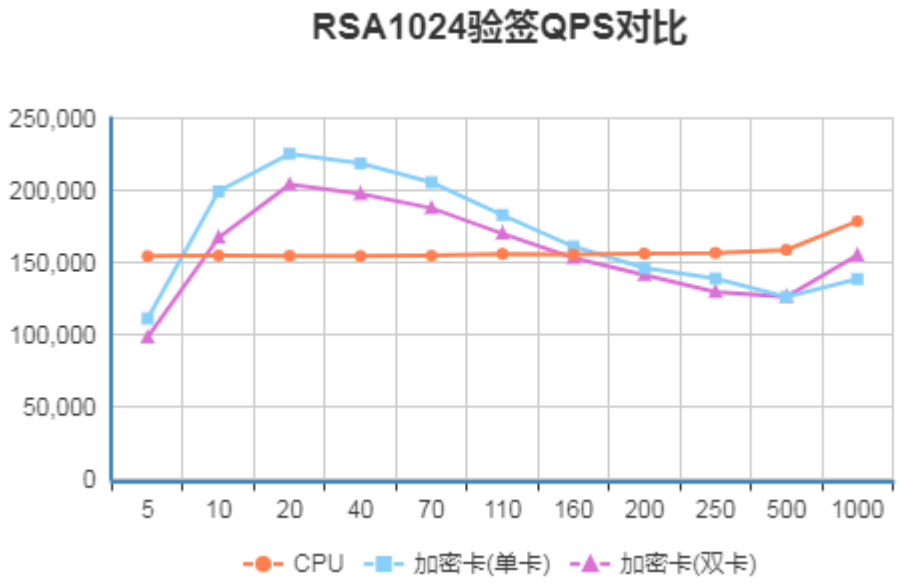
摘要算法和 AES 算法多进程测试情况下，加密卡和 CPU 性能上同样基本一致，这里不再单独列出结果。

主要统计的是 RSA 算法和 ECDSA 算法在不同并发情况下的 QPS 数据，测试采用的并发数分别为 5、10、20、40、70、110、160、200、250、500、1000,并分别记录不同算法的速度，进行对比显示。

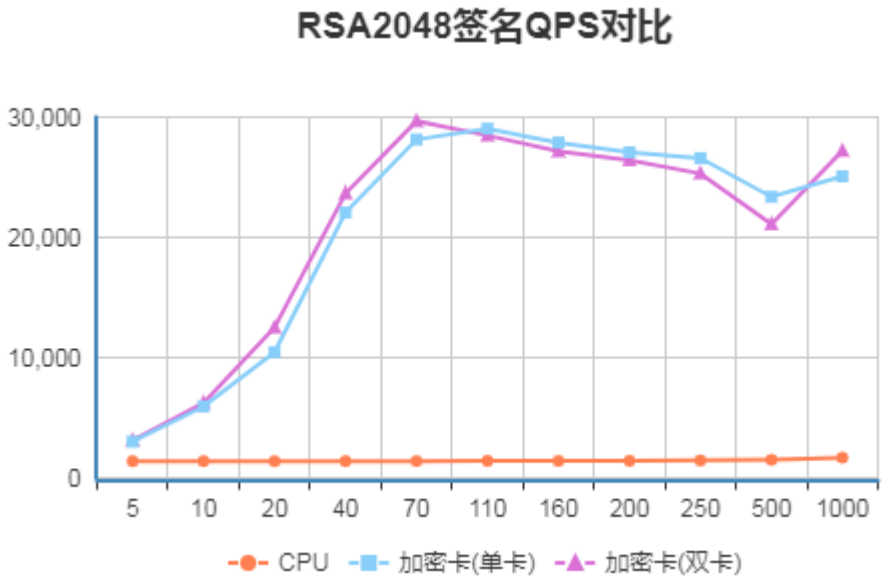
同时加入双卡测试对比条件，也就是在主机 1 支持的情况下插入两张加密卡进行测试，根据主机一的卡槽信息，分别为 PCIe X16 和 PCIe X1 Edge Free 两个。

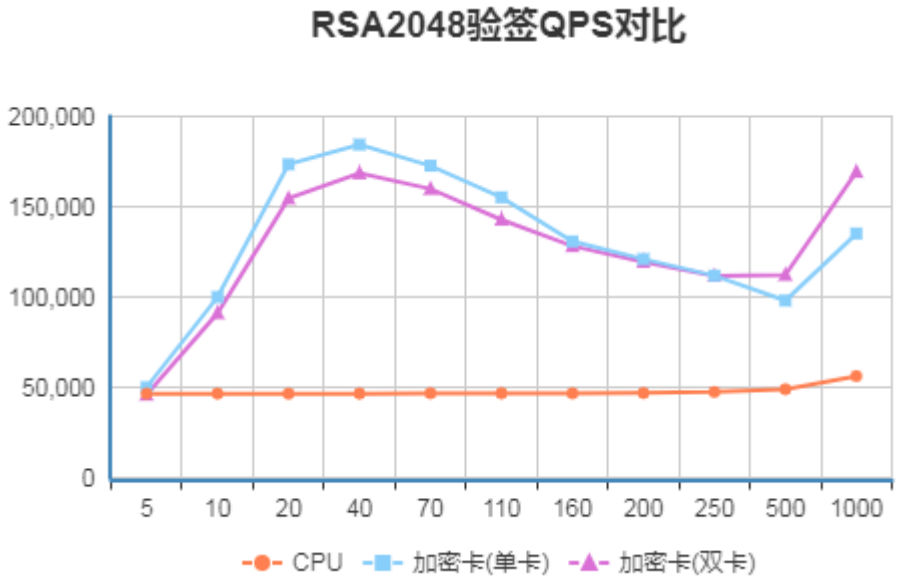
6.2.2.1. RSA1024



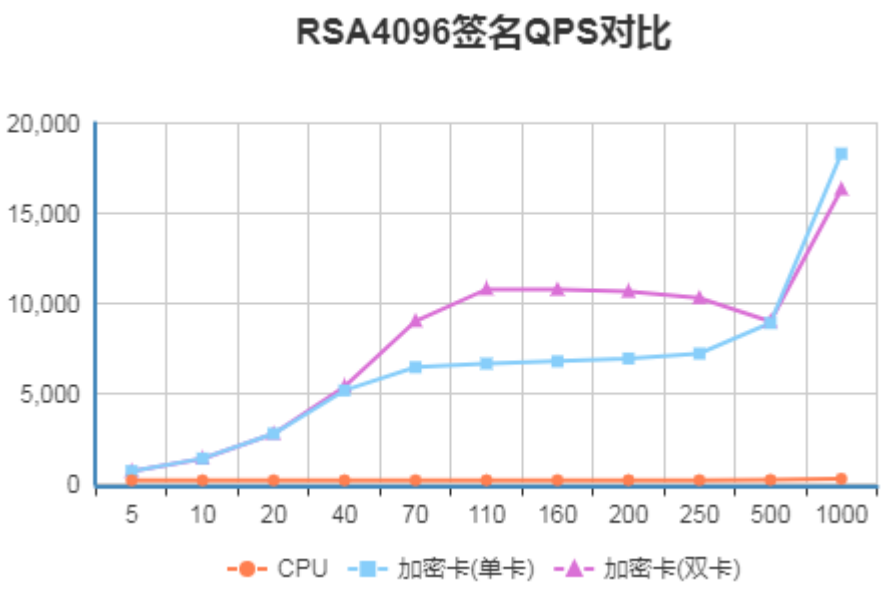


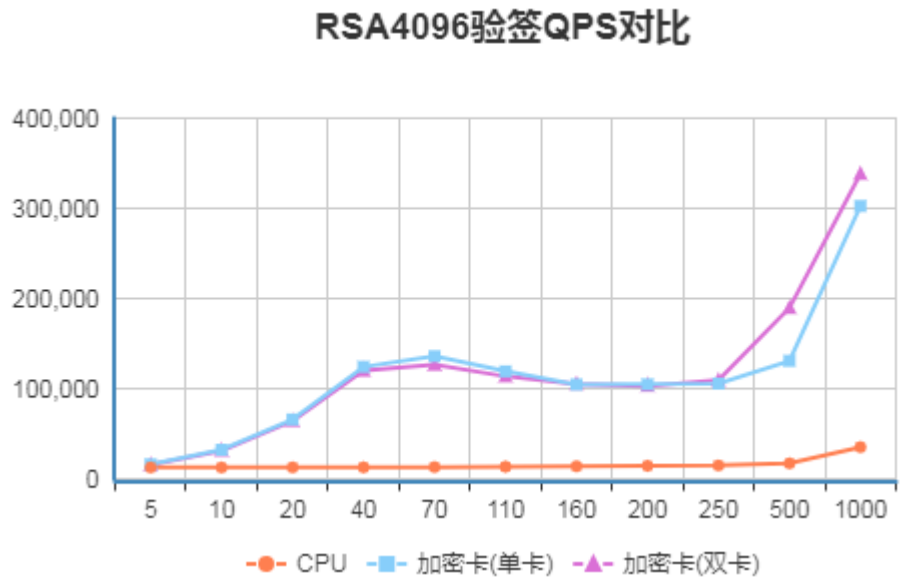
6.2.2.2. RSA2048



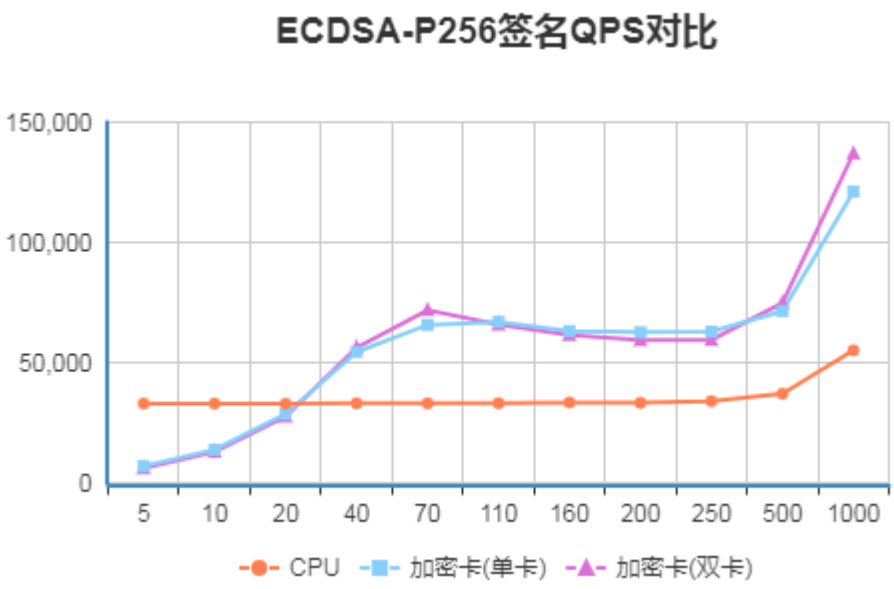


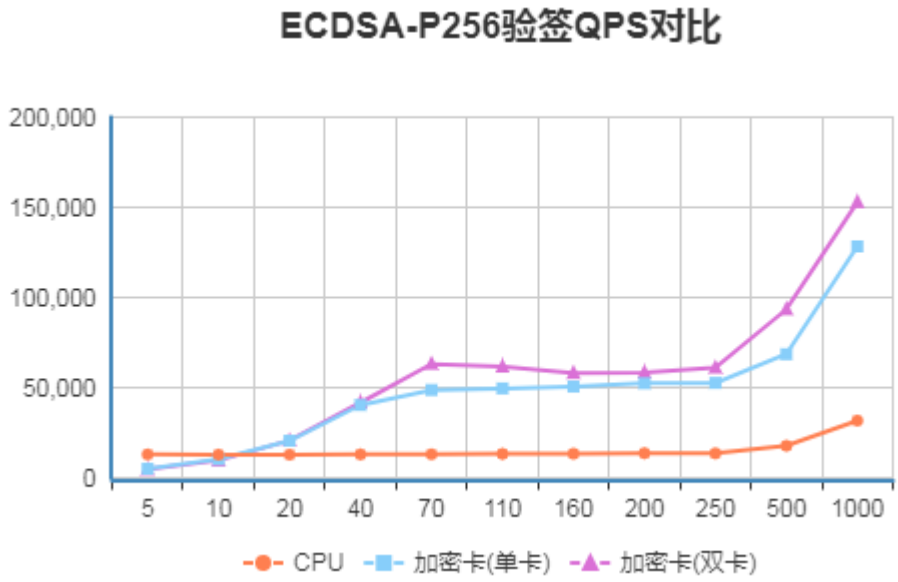
6.2.2.3. RSA4096





6.2.2.4. ECDSAP256





6.2.3. 测试总结

通过上述测试结果的分析，可以看出摘要算法和 AES 算法 CPU 和加密卡的性能基本一致；

而在测试 RSA 和 ECDSA 算法时，CPU 测的结果，比较稳定，变化不大，在并发数比较低的情况下，二者差距不大，但是在并发数超过一定值（多为 30）时，加密卡有较大的提升。

同时，在该主机测多卡时，并没有得到预期的结果，无法对多卡性能进行评估，双卡与单卡在大部分时间的性能保持一致。

6.3. 基于主机二的测试结果（性能测试）

6.3.1. 单进程测试

表 1 CPU 计算结果

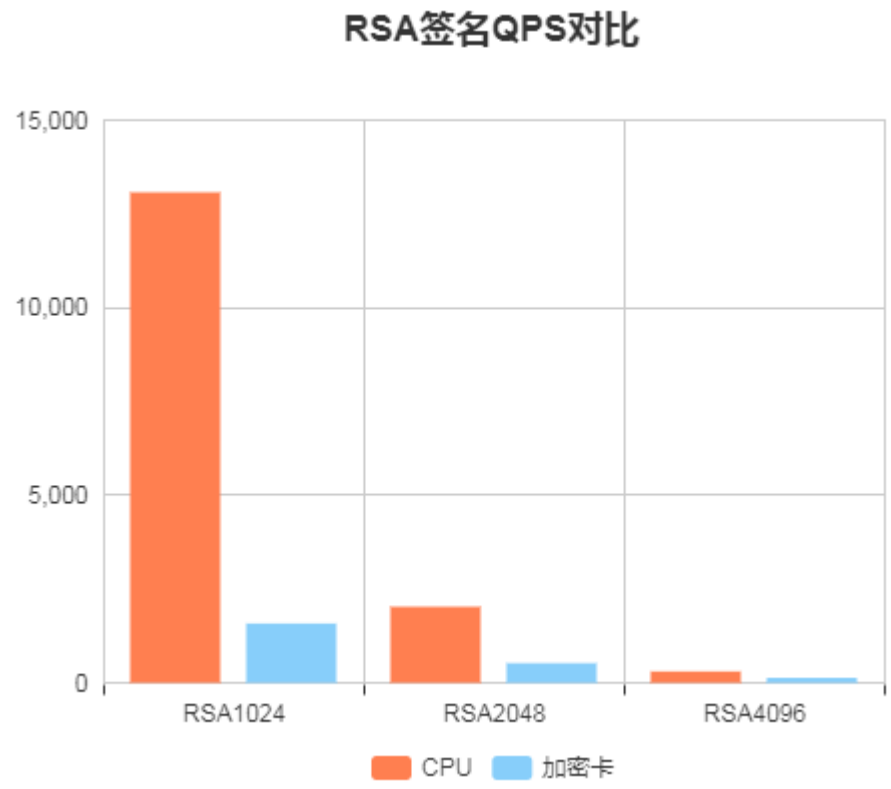
类型	16bytes	64bytes	256bytes	1024bytes	8192bytes
sha1	35.989	126.274	384.641	783.653	1122.047
sha256	96.888	218.674	403.741	505.625	544.145
sha512	67.940	271.526	458.594	688.925	805.683
md5	33.108	114.366	326.685	590.452	804.691
aes-128-cbc	168.676	186.657	191.129	192.746	193.306

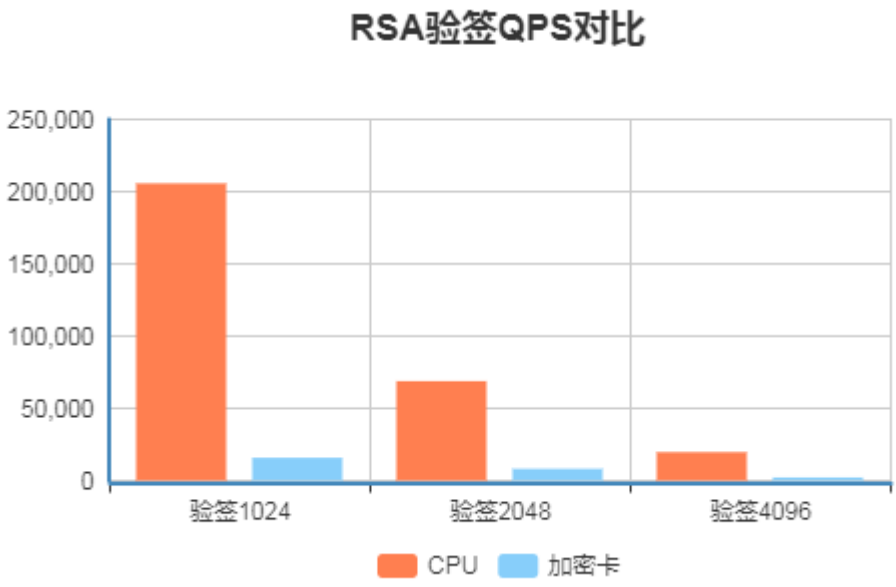
表 2 加密卡计算结果

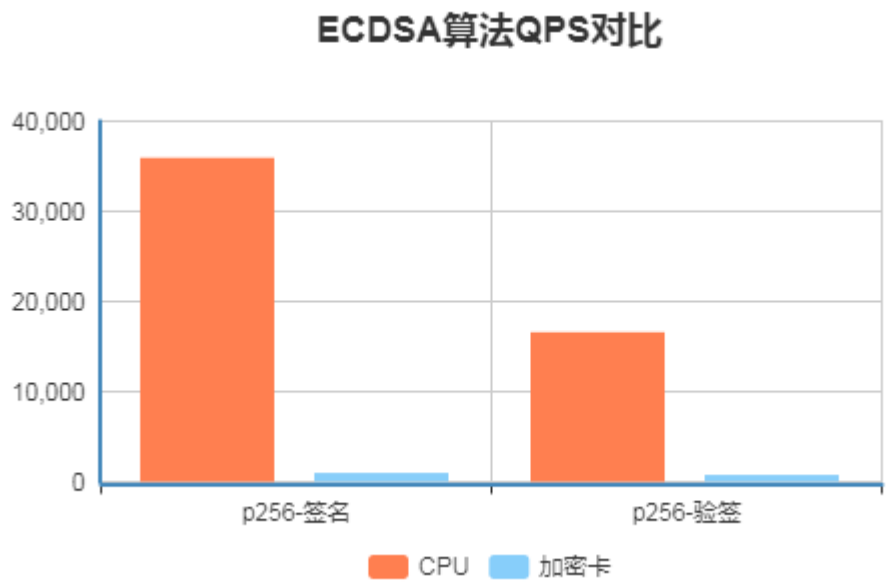
类型	16bytes	64bytes	256bytes	1024bytes	8192bytes
sha1	35.781	125.472	381.946	782.652	1124.851
sha256	96.947	218.749	404.041	505.709	544.964
sha512	67.909	271.629	462.112	688.880	806.029
md5	33.348	115.774	327.255	591.599	804.809
aes-128-cbc	168.425	186.576	191.009	192.821	193.536

通过上面 2 张表格的对比，可以发现这几种算法的性能在使用 CPU 和加密卡的时候基本没有变化。

下面几幅图为基于主机二测试的结果，测试为单卡、单进程测试条件对比，包括 rsa（1024，2048，4096）和 ecdsa（p256）算法。


RSA签名QPS对比
15,000
10,000
5,000
0
RSA1024
RSA2048
RSA4096
CPU
加密卡


RSA验签QPS对比
250,000
200,000
150,000
100,000
50,000
0
验签1024
验签2048
验签4096
CPU
加密卡



通过上面两幅图可以看出在单进程测试条件下，加密卡的性能并没有发挥出来，不管是 RSA 还是 ECC 的算法都要比软件执行速度要慢。

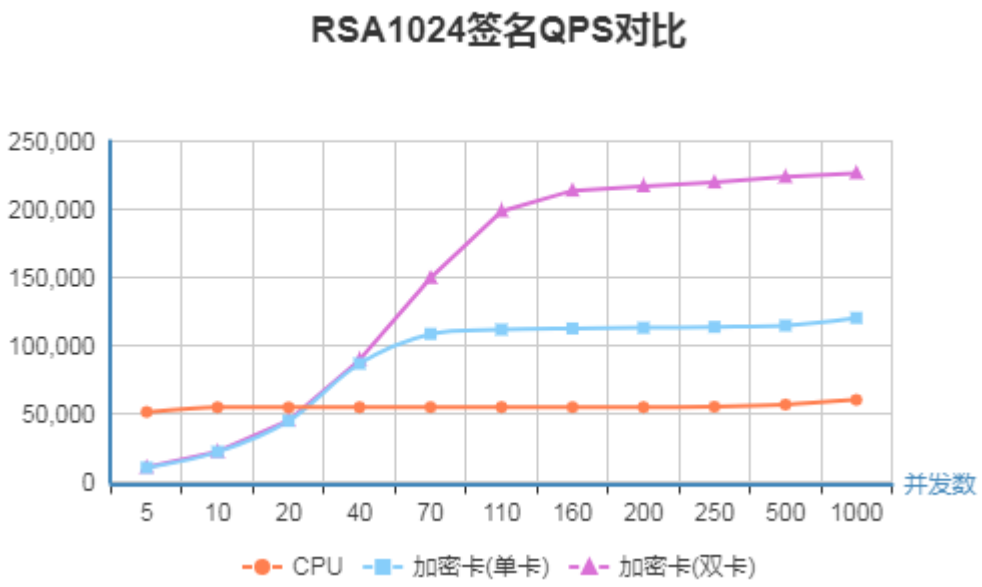
6.3.2. 多进程测试

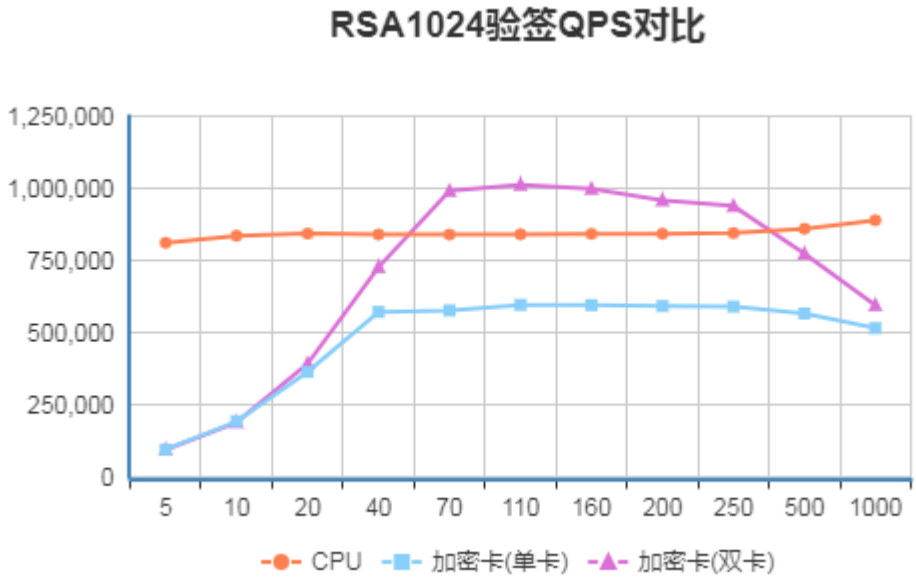
摘要算法和 AES 算法多进程测试情况下，加密卡和 CPU 性能上同样基本一致，这里不再单独列出结果。

同样测试并发数与主机一一致。测试采用的并发数分别为 5、10、20、40、70、110、160、200、250、500、1000

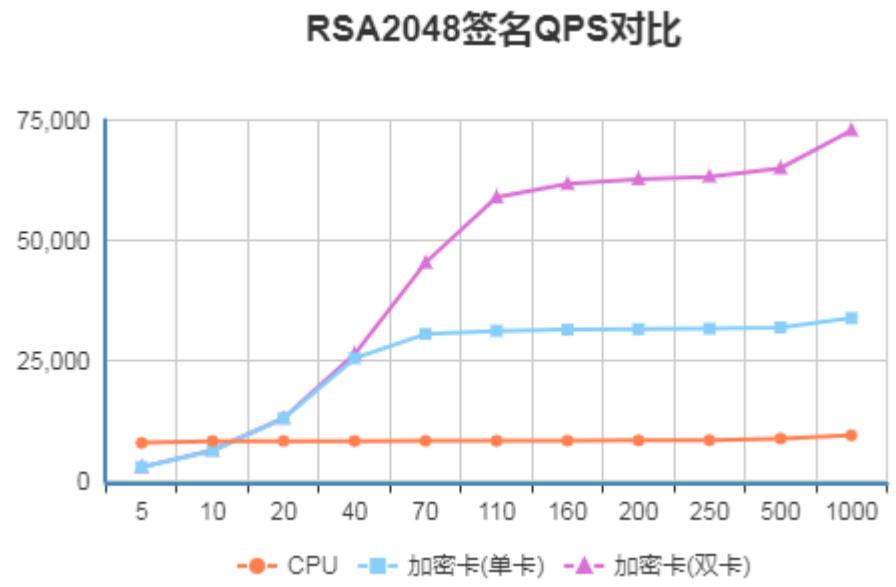
同时加入双卡测试，也就是在主机 2 支持的情况下插入两张加密卡进行测试，根据主机一的卡槽信息，分别为 PCIe X16 和 PCIe X16 两个。

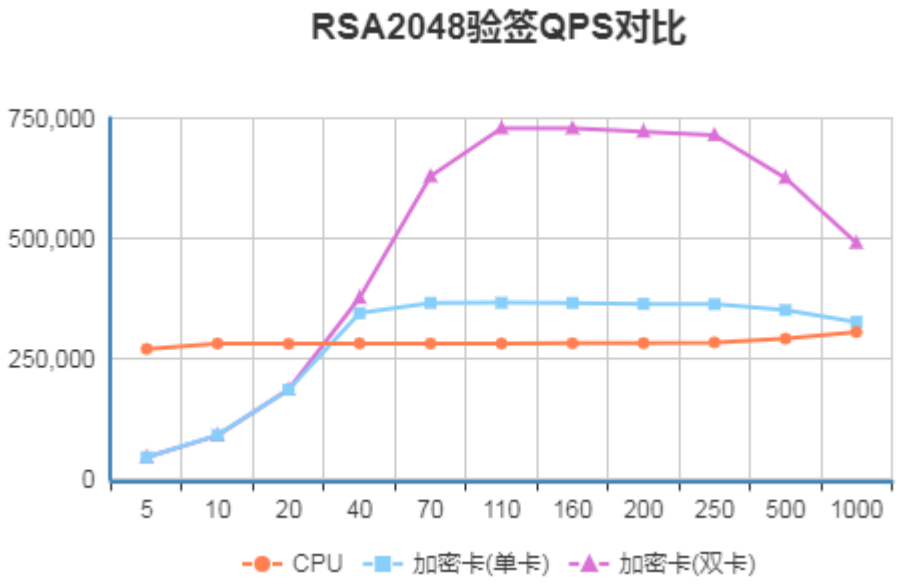
6.3.2.1. RSA1024



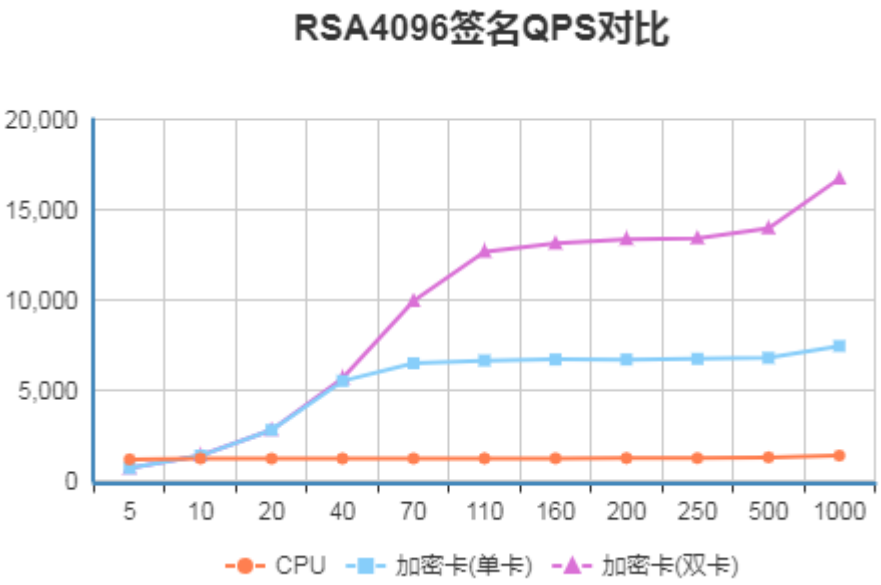


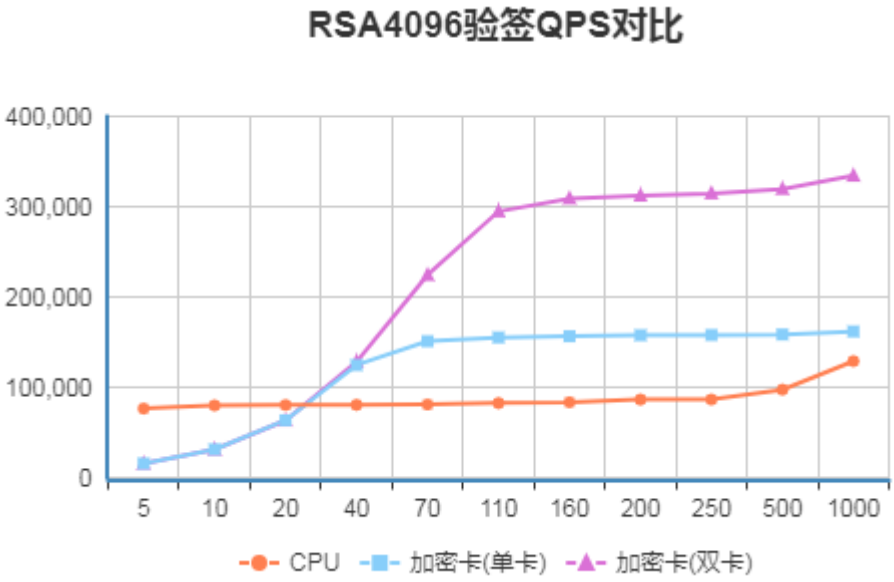
6.3.2.2. RSA2048



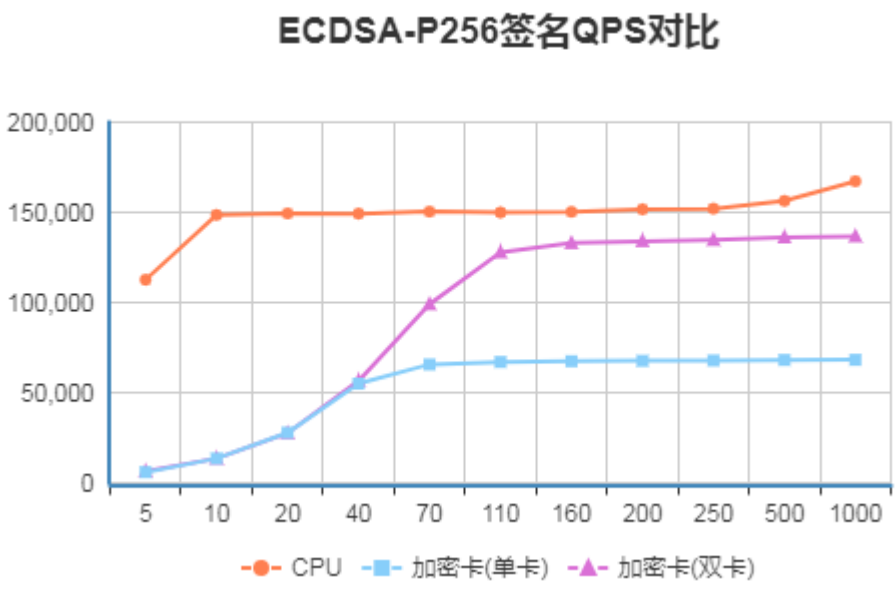


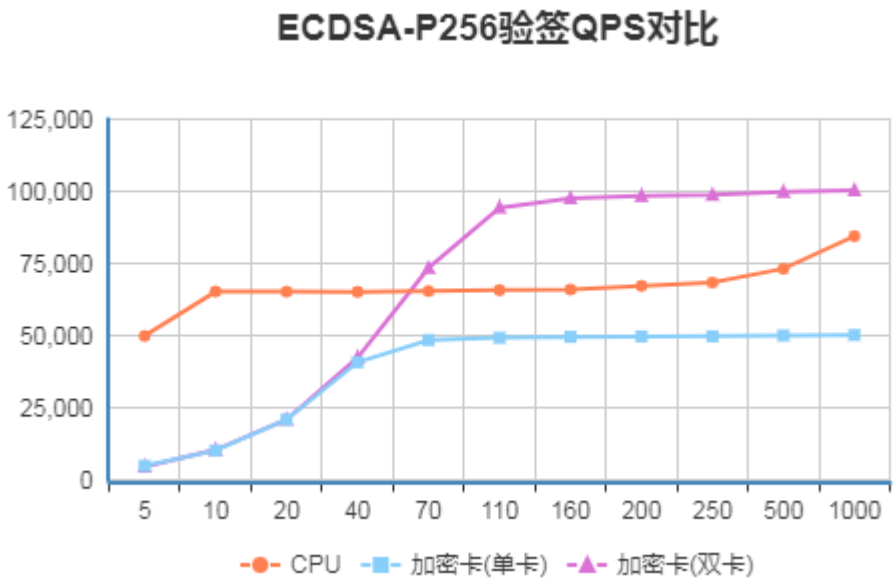
6.3.2.3. RSA4096





6.3.2.4. ECDSAP256





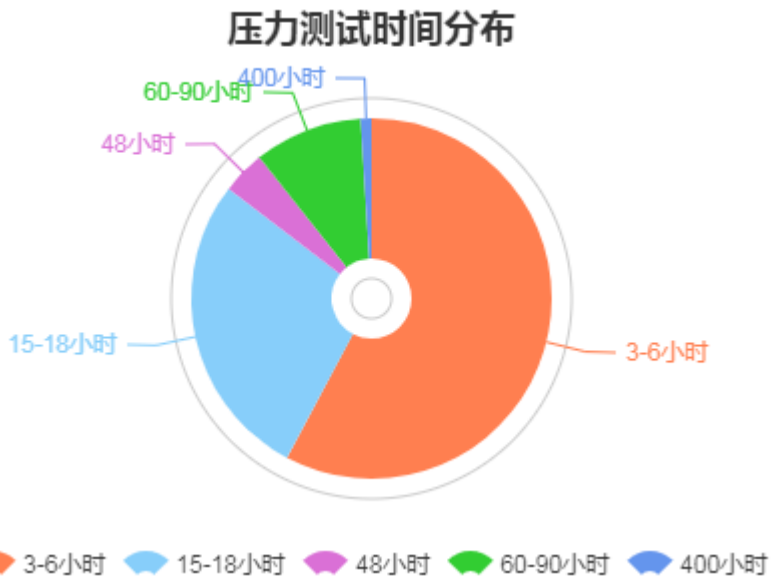
6.3.2.5. 测试总结

摘要算法和 AES 算法 CPU 和加密卡测出的速度和性能基本一致，没有大的变化，而 RSA 和 ECDSA 算法则有较大的不同。

同时将上述图跟主机一得到结果进行对比可得，算法的性能跟主机的性能也有很大关系，在主机二性能比较高的情况下，一些计算量比较小的反而 CPU 跑得更快，例如 rsa1024，ecdsa 等算法。而 rsa2048，rsa4096 在并发数提升到一定程度（通常为 30 左右）时，性能可以得到成倍的提升。同时，双卡的功能也收到了很好的预期效果，峰值基本为单卡时的两倍。

6.4. 疲劳测试

疲劳测试为针对所有已发行的板子进行，使用 openssl 的接口开发对应的测试程序，使加密卡一直处于高负荷工作状态，并统计无故障运行时间。测试程序同时覆盖了 rsa（1204，2048，4096）算法的加解密、签名验签和 ecdsa（p256k1，p256r1）算法的签名验签操作，并采用多线程运行机制，并发数设为 150。下图为该批次加密卡的测试无故障运行时间的分布图。



待测加密卡为 200 张，鉴于时间的关系，57%的卡测试时长为 3-6 个小时，27%的卡测试时长为 15-18 个小时，15%的卡测试时长为 40-90 个小时， 1%的卡测试时长超过 400 个小时。