|  |  |
| --- | --- |
| 产品名称Product name | 密级Confidentiality level |
| 加密卡 |  |
| 产品版本Product version | Total 12 pages共12页 |
| V1.1.1 |  |

SSC区块链运算加密卡用户使用手册

|  |  |  |  |
| --- | --- | --- | --- |
| 拟制: | 马亚飞 | 日期： | 2018.4.23 |
| 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

版权所有 侵权必究

修订记录

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 修订版本 | 描述 | 作者 |
| 2018年4月20日 | 1.0.0 | 初稿 | 陈金强 |
| 2018年4月23日 | 1.1.0 | 添加上层用户使用说明 | 马亚飞 |
| 2018年4月25日 | 1.1.1 | 补充用户使用手册 | 马亚飞 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目录

[**1.** **概述** 4](#_Toc512504264)

[**2.** **简介** 4](#_Toc512504265)

[**2.1.** **结构** 4](#_Toc512504266)

[**2.2.** **特性** 4](#_Toc512504267)

[**2.3.** **性能** 5](#_Toc512504268)

[**3.** **加密卡构成** 5](#_Toc512504269)

[**3.1.** **硬件** 5](#_Toc512504270)

[**3.2.** **软件** 6](#_Toc512504271)

[**4.** **安装和配置** 6](#_Toc512504272)

[**4.1.** **概述** 6](#_Toc512504273)

[**4.2.** **硬件安装** 6](#_Toc512504274)

[**4.3.** **驱动安装** 7](#_Toc512504275)

[**5.** **加密卡的使用** 8](#_Toc512504276)

[**6.** **引擎编译** 9](#_Toc512504277)

[**6.1.** **文件补丁** 9](#_Toc512504278)

[**6.2.** **openssl编译** 10](#_Toc512504279)

[**6.3.** **引擎动态库编译** 10](#_Toc512504280)

[**7.** **常见问题解答** 11](#_Toc512504281)

[**附录A 加密卡拨码开关配置** 12](#_Toc512504282)

1. **概述**

SSC区块链运算加密卡是深圳数字电视国家工程实验室开发的高性能低功耗PCIe接口加密卡，提供密码运算加速以及密钥安全管理功能。可应用于网络安全应用，区块链，数据中心等领域。本文档详细描述了加密卡的功能、使用说明及注意事项。

1. **简介**

加密卡基于NXP的C293PCIe芯片开发研制，内置硬件加速处理引擎，支持RSA、ECDSA等非对称算法和AES，SHA类对称算法。加密卡采用PCIe接口，作为单卡的主机系统连接的示意图如下图1所示：

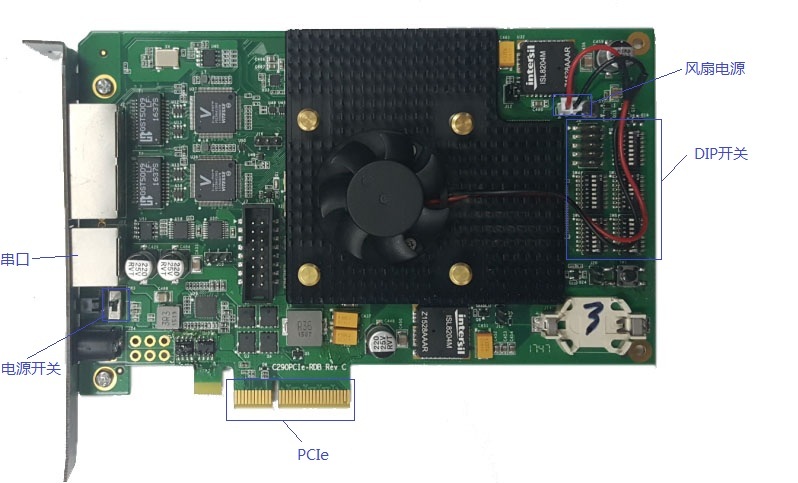


*图1 工作示意图*

加密卡与主机相连后可以作为一个完整的主机对外提供高性能的应用服务。

* 1. **结构**

加密卡硬件结构如下图所示：



*图2 加密卡板子布局*

* 1. **特性**
* 内置3个安全引擎，每个引擎有15个公钥硬件加速器；
* NIST认证的随机数发生器；
* RSA（最大支持4096-bit）签名验签算法。
* ECC 密码算法（素数域最大 1024 位长，二元域最大 1024 位长）；
* 支持 PCIE X1、X2 、X4 模式；
* 四通道的DMA控制器；
* PCIe 的连接器规格： Edge Free / Multi-lane PCIe X1 Connector : Support X1、 X2、 X4；
* PCIe电源：不低于40瓦的供电电源，避免电源供电不足出现异常；
* 功耗：加密卡功耗在30瓦左右。
* 工作条件：
  + 工作环境温度：0℃-45℃；
  + 工作环境相对湿度：45％-75％；
  + 存贮环境温度：0℃-80℃；
  + 存贮环境相对湿度：20％-95％。
  + 可靠性指标：MTBF 大于等于 20000 小时。
  1. **性能**

下表性能指标采用的是NXP提供的内核测试工具进行统计的结果，测试结果与主机的性能有关：

|  |  |  |
| --- | --- | --- |
| **测试主机** | **Intel(R) Pentium(R) CPU G840 @ 2.80GHz** | |
|  | 性能 | 执行命令 |
| **RSA2048 Public** | 331125.82次/s | c29x\_driver\_perf\_profile.sh RSA\_PUB\_OP\_2K -m 0x2 -t 1 -s 10 |
| **RSA2048 Private** | 31695.72次/s | c29x\_driver\_perf\_profile.sh RSA\_PRV\_OP\_2K -m 0x2 -t 1 -s 10 |
| **ECDSAP256 sign** | 90826.52次/s | c29x\_driver\_perf\_profile.sh ECDSA\_SIGN\_TEST -m 0x2 -t 1 -s 10 |
| **ECDSAP256 verify** | 67204.30次/s | c29x\_driver\_perf\_profile.sh ECDSA\_VERIFY\_TEST -m 0x2 -t 1 -s 10 |

1. **加密卡构成**
   1. **硬件**

* 加密卡一张。
  1. **软件**
* 加密卡驱动程序一份，包括四个文件（***fsl\_pkc\_crypto\_offload\_drv.ko****、****cryptodev.ko****、****pkc-firmware.bin****、****crypto.cfg***）；
* 加密卡依赖openssl源码一份（***openssl\_1.0.2g***）；
* 加密卡动态引擎库一份（***libeng\_cryptodev.so***）。

1. **安装和配置**
   1. **概述**

加密卡工作的框图如下所示：



*图3 加密卡工作框图*

加密卡的安装主要分为硬件安装和驱动安装两部分。硬件安装即将加密卡正确的安装  
到主板上的 PCIe 插槽中，其间需要注意主板是否具有相应的 PCIe 插槽。在正确安装  
加密卡后，运行相应的驱动程序，并按照流程进行安装。以下对安装过程中的注意事  
项进行介绍说明。

* 1. **硬件安装**

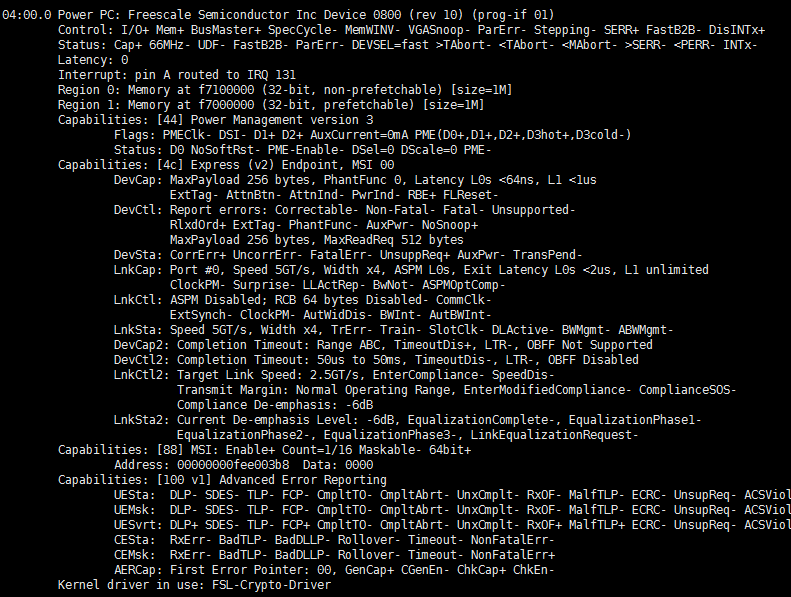
加密卡外形采用 PCIE X4 接口，实际上加密卡支持 X1、 X2、 X4 协议。加密卡由精  
密的算法芯片及其它元器件构成，这些集成电路很容易受到静电影响而损坏，所以在安装前  
做好以下准备：

1. 请将主机的电源关闭，并且拔除电源插头；
2. 将主板中的电源插座上的插头拔除时，请确认电源的开关是关闭状况；
3. 拿取密码卡时请尽量避免芯片焊接部分，且最好能够戴上防静电手套；
4. 确保主板具有相应匹配的 PCIe 插槽。

在确保以上内容正确无误后，可按以下步骤安装加密卡：

1. 从机箱后壳上移除对应 PCIe 插槽上的扩充挡板及螺丝；
2. 将加密卡小心的对准 PCIe 插槽并且很确实的插入 PCIe 插槽中(注意：务必确认将卡上的金手指的金属触点与 PCIe 插槽接触在一起)；
3. 用螺丝刀将螺丝锁上使加密卡固定在机箱壳上；
4. 确认无误后，重新开启电源，即完成加密卡的硬件安装。
5. 如需添加多卡支持，需确保主机主板支持多个PCIe插槽，为确保性能最好都兼容PCIe X4插槽；若为PCIe X1插槽，需要使用Edge Free，不然无法正常插入。
   1. **驱动安装**

在确认加密卡硬件安装成功后，可以运行“sudo lspci”或“sudo lspci –vvv”命令来观察是否新设备插入。如果没有发现设备，需要检查一下加密卡的硬件配置（参考3.2.1章节，尤其是拨码开关的SW7的第一位是否开启）或者加密卡是否插牢。如下图所示：



*图4 lspci命令执行结果图*

检查驱动安装所需文件是否完整，然后按照下列步骤执行：

**步骤1**:创建/etc/crypto目录，并拷贝crypto.cfg和pkc-firmware.bin文件到/etc/crypto目录中，具体命令为：

|  |
| --- |
| $ sudo mkdir /etc/crypto |
| $ sudo cp crypto.cfg /etc/crypto/crypto.cfg |
| $ sudo cp pkc-firmware.bin /etc/crypto/pkc-firmware.bin |

**步骤2**:加载驱动文件，具体命令为：

|  |
| --- |
| $ sudo insmod fsl\_pkc\_crypto\_offload\_drv.ko |
| $ sudo insmod cryptodev.ko |

**步骤3**:使用lsmod命令或者dmesg命令查看内核日志来检查驱动是否加载成功。此外加密卡驱动支持多卡处理，只需PCIe口兼容即可在同一台主机上插入多张加密卡，无须进行额外的配置。

1. **加密卡的使用**

加密卡采用openssl的动态引擎机制进行实现，依赖的openssl版本为1.0.2g，编译出的引擎库文件为libeng\_cryptodev.so，详细的编译选项和代码参照引擎编译章节。

由于引擎有部分代码并非openssl公共代码，因此引擎的加载需要依赖openssl编出的库文件libcrypto.so和libssl.so,使用export命令添加环境变量，例如：

|  |
| --- |
| $ export LD\_LIBRARY\_PATH=/home/动态库所在目录/ |

然后就可以按照动态引擎的加载机制进行加载了，例如通过命令行的方式进行加载（这里使用的是根据代码编译出来的openssl可执行程序）：

|  |
| --- |
| $ ./openssl speed -engine libeng\_cryptodev.so rsa2048  engine "cryptodev" set.  Doing 2048 bit private rsa's for 10s:  … |

出现“engine “cryptodev” set”输出表示引擎已经成功加载。代码调用的方式如下：

*ENGINE\_load\_dynamic();*

*ENGINE \*cryptodev = ENGINE\_by\_id("dynamic");*

*ENGINE\_ctrl\_cmd\_string(cryptodev, "ID", "cryptodev", 0);*

*ENGINE\_ctrl\_cmd\_string(cryptodev, "SO\_PATH", "/xxxx/libeng\_cryptodev.so", 0);*

*ENGINE\_ctrl\_cmd\_string(cryptodev, "LOAD", NULL, 0);*

*int init\_ret = ENGINE\_init(cryptodev);*

*int set\_ret = ENGINE\_set\_default\_RSA(cryptodev);*

具体的使用方式要参考openssl的引擎api，参考链接为：

<https://www.openssl.org/docs/man1.0.2/crypto/engine.html>

参考命令：

./openssl speed –elapsed –multi 50 -engine ./libeng\_cryptodev.so rsa2048

./openssl speed –elapsed –multi 50 -engine ./libeng\_cryptodev.so ecdsap256

1. **引擎编译**
   1. **文件补丁**

加密卡引擎的编译需要先编译openssl，采用的openssl版本为1.0.2g，为了使openssl支持动态引擎机制，需要对openssl的配置进行相应的修改，同时补丁文件中还包括一些bug的修复。具体修改文件如下所示($OPENSSL表示openssl源码的根目录)：

1. **$OPENSSL/openssl.ld文件**:

*@@ -4622,6 +4622,9 @@ OPENSSL\_1.0.2d {*

*ECDSA\_sign\_hardware;*

*RSA\_public\_encrypt\_software;*

*RSA\_private\_decrypt\_software;*

*+               bind\_engine;*

*+               ecdsa\_check;*

*+               ENGINE\_\*;*

*local:*

*\*;*

*};*

1. **$OPENSSL/crypto/engine/Makefile:**

*@@ -21,14 +21,14 @@ LIBSRC= eng\_err.c eng\_lib.c eng\_list.c eng\_init.c eng\_ctrl.c \*

*eng\_table.c eng\_pkey.c eng\_fat.c eng\_all.c \*

*tb\_rsa.c tb\_dsa.c tb\_ecdsa.c tb\_dh.c tb\_ecdh.c tb\_rand.c tb\_store.c \*

*tb\_cipher.c tb\_digest.c tb\_pkmeth.c tb\_asnmth.c \*

*-       eng\_openssl.c eng\_cnf.c eng\_dyn.c eng\_cryptodev.c \*

*-       eng\_rdrand.c interface\_crypfun.c*

*+       eng\_openssl.c eng\_cnf.c eng\_dyn.c  \*

*+       eng\_rdrand.c*

*LIBOBJ= eng\_err.o eng\_lib.o eng\_list.o eng\_init.o eng\_ctrl.o \*

*eng\_table.o eng\_pkey.o eng\_fat.o eng\_all.o \*

*tb\_rsa.o tb\_dsa.o tb\_ecdsa.o tb\_dh.o tb\_ecdh.o tb\_rand.o tb\_store.o \*

*tb\_cipher.o tb\_digest.o tb\_pkmeth.o tb\_asnmth.o \*

*-       eng\_openssl.o eng\_cnf.o eng\_dyn.o eng\_cryptodev.o \*

*-       eng\_rdrand.o interface\_crypfun.o*

*+       eng\_openssl.o eng\_cnf.o eng\_dyn.o \*

*+       eng\_rdrand.o*

1. **$OPENSSL/crypto/engine/eng\_int.h**

*@@ -175,7 +175,7 @@ struct engine\_st {*

*const ENGINE\_CMD\_DEFN \*cmd\_defns;*

*int flags;*

*/\* reference count on the structure itself \*/*

*-    int struct\_ref;*

*+    unsigned long struct\_ref;*

1. **$OPENSSL/crypto/engine/eng\_lib.c**

*@@ -71,7 +71,7 @@ void engine\_set\_all\_null(ENGINE \*e)*

*int engine\_free\_util(ENGINE \*e, int locked)*

*{*

*-    int i;*

*+    unsigned long i;*

*if (e == NULL)*

*return 1;*

* 1. **openssl编译**

1. 首先将cryptodev.h头文件拷贝到/usr/include/crypto目录下；
2. 在openssl目录下，执行./config -DENGINE\_DYNAMIC\_SUPPORT -shared && make depend && make来进行配置和编译；
   1. **引擎动态库编译**
3. 如需编译引擎库，需获取引擎库代码，然后放入$OPENSSL/crypto/engine目录中；
4. 进入到$OPENSSL/crypto/engine目录；
5. 编译生成目标文件：

*gcc -Wall  -I.. -I../.. -I../modes -I../asn1 -I../evp -I../../include  -fPIC -DOPENSSL\_PIC -DOPENSSL\_THREADS -D\_REENTRANT -DDSO\_DLFCN -DHAVE\_DLFCN\_H -DENGINE\_DYNAMIC\_SUPPORT -Wa,--noexecstack -m64 -DL\_ENDIAN -O3 -Wall -DOPENSSL\_IA32\_SSE2 -DOPENSSL\_BN\_ASM\_MONT -DOPENSSL\_BN\_ASM\_MONT5 -DOPENSSL\_BN\_ASM\_GF2m -DSHA1\_ASM -DSHA256\_ASM -DSHA512\_ASM -DMD5\_ASM -DAES\_ASM -DVPAES\_ASM -DBSAES\_ASM -DWHIRLPOOL\_ASM -DGHASH\_ASM -DECP\_NISTZ256\_ASM -DHAVE\_CRYPTODEV -c -o eng\_cryptodev.o eng\_cryptodev.c*

1. 链接生成动态库：

*gcc -shared eng\_cryptodev.o -o libeng\_cryptodev.so -lcrypto -L../../*

1. **常见问题解答**

在加密卡的安装使用过程中可能出现的问题和相应解决办法：

1. 在安装完成加密卡的硬件后，如若未发现设备，请确认以下环节：
   1. 加密卡的安装过程是否正确进行；
   2. 加密卡与主机的连接是否紧密，可重新拔插加密卡，并确认加密卡与PCIe插槽的紧密连接；
   3. 拨码开关是否与默认配置保持一致；
   4. 加密卡是否支持主机的操作系统。
2. 在安装加密卡驱动后，如果提示加载失败，请检查以下环节：
   1. 是否安装正确加密卡驱动程序；
   2. 确认操作系统是否匹配。
   3. 若错误仍存在，与我司技术员联系。
3. 加密卡损坏：
   1. 如在安装之前发现加密卡器件脱落、破损或明显的损坏情况，应立刻停止安装，与我司技术员联系。
   2. 如在上电工作中，出现加密卡器件击穿损毁的情况，应立即断电，将加密卡取下，与我司技术员联系。

**附录A 加密卡拨码开关配置**

加密卡支持多种工作模式，通过DIP拨码开关进行配置，拨码开关的位置在图1中为DIP开关蓝色边框内的部分，具体分布为：

|  |  |
| --- | --- |
| 两排针插座 | SW5 |
| SW4 | SW7 |
| SW6 | SW8 |

下图为拨码开关的配置，加密卡已经配置为主机工作模式，用户无须再进行配置（ON为0， OFF为1）。

|  |  |  |
| --- | --- | --- |
| SW4:0101 1000 | SW5:1111 1000 | SW6:0000 1111 |
| |  |  |  |  | | --- | --- | --- | --- | | 1 |  | 0 | O | | 2 | 1 |  | N | | 3 |  | 0 |  | | 4 | 1 |  |  | | 5 | 1 |  |  | | 6 |  | 0 |  | | 7 |  | 0 |  | | 8 |  | 0 |  | | |  |  |  |  | | --- | --- | --- | --- | | 1 | 1 |  | O | | 2 | 1 |  | N | | 3 | 1 |  |  | | 4 | 1 |  |  | | 5 | 1 |  |  | | 6 |  | 0 |  | | 7 |  | 0 |  | | 8 |  | 0 |  | | |  |  |  |  | | --- | --- | --- | --- | | 1 |  | 0 | O | | 2 |  | 0 | N | | 3 |  | 0 |  | | 4 |  | 0 |  | | 5 | 1 |  |  | | 6 | 1 |  |  | | 7 | 1 |  |  | | 8 | 1 |  |  | |
| SW7:0000 0111 | SW8:0000 0001 |  |
| |  |  |  |  | | --- | --- | --- | --- | | 1 |  | 0 | O | | 2 |  | 0 | N | | 3 |  | 0 |  | | 4 |  | 0 |  | | 5 |  | 0 |  | | 6 | 1 |  |  | | 7 | 1 |  |  | | 8 | 1 |  |  | | |  |  |  |  | | --- | --- | --- | --- | | 1 |  | 0 | O | | 2 |  | 0 | N | | 3 |  | 0 |  | | 4 |  | 0 |  | | 5 |  | 0 |  | | 6 |  | 0 |  | | 7 |  | 0 |  | | 8 | 1 |  |  | |  |

*图5 拨码开关默认配置*