TrustZone

部署指南

文档版本 02

发布日期 2022-01-19

华为技术有限公司

商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

目 录

1 介绍

TrustZone是ARM架构下为解决可能遇到的软硬件安全问题提出的一种硬件解决方案。iTrustee基于TrustZone技术实现了整套安全解决方案，包含正常模式的客户端应用（Client Application，CA）、安全模式的可信应用（Trusted Application，TA）、安全模式下的可信操作系统。

本文档主要介绍了：

- 部署iTrustee TA（Trusted Application）和CA（Client Application）应用开发环境。
- 在已集成TrustZone特性的TaiShan服务器上部署、运行TA/CA应用。

2 TA/CA 应用开发环境搭建

2.1 环境要求

TA/CA应用开发环境对硬件、系统版本无特殊要求，只需要基于iTrustee SDK套件开发TA/CA应用即可。相关软件包获取见**表2-1**。

表 2-1 软件包获取

软件包名称	说明	获取途径
iTrustee SDK	TA/CA应用开发者套件	**https://gitee.com/openeuler/itrustee_sdk**
libboundscheck	华为安全函数库	**https://gitee.com/openeuler/libboundscheck**
itrustee_client	iTrustee REE侧patch包源码	**https://gitee.com/openeuler/itrustee_client**
rsa-demo	Demo示例代码	签署免责协议，定向提供

说明

其中iTrustee SDK软件包后续会有更新， 因时延未能及时同步至开源社区，如有需要请与华为接口人确认并获取最新版本。通过此方式获取最新iTrustee SDK需同华为业务负责人签订免责协议。

2.2 搭建步骤

本文以iTrustee SDK和rsa-demo代码为例介绍TA/CA编译环境部署，系统版本为OpenEuler 20.03 LTS-SP1。

获取 iTrustee patch 包

步骤1 获取itrustee_client源码。

下载地址：**https://gitee.com/openeuler/itrustee_client**

步骤2 获取libboundscheck源码。

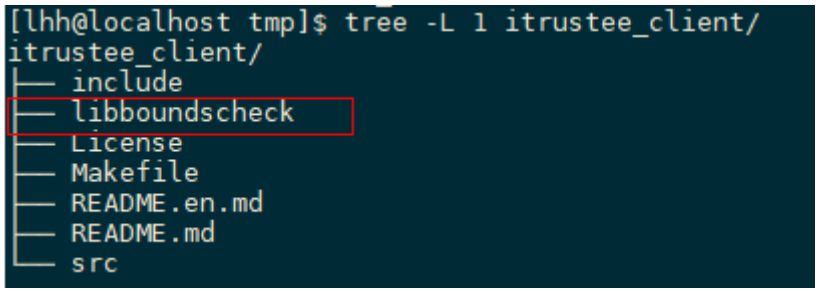
下载地址：**https://gitee.com/openeuler/libboundscheck**

步骤3 安装编译依赖。

```
yum install openssl-devel zlib-devel
```

步骤4 编译itrustee_client。

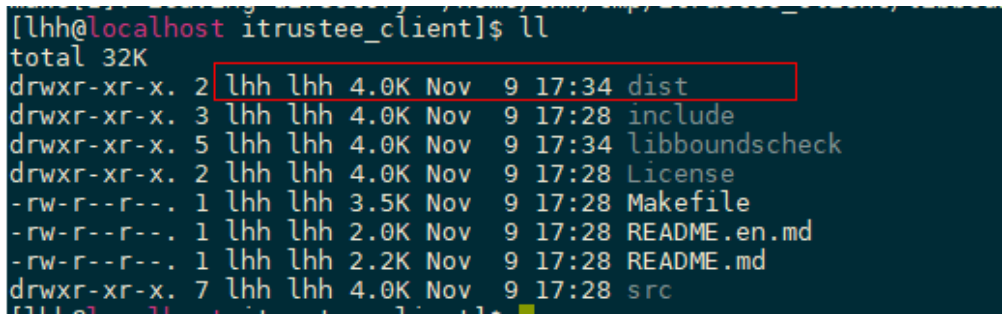
1. 将libboundscheck源码放置在itrustee_client/目录下，并修改文件夹名为libboundscheck。层级目录如下图所示。

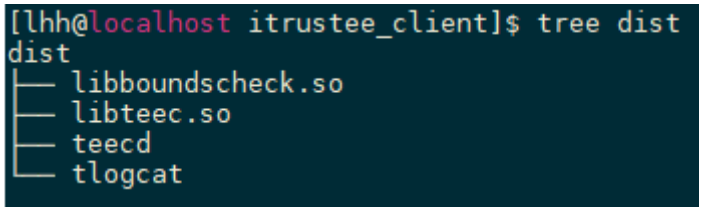


2. 编译。

```
cd itrustee_client && make
```

编译成功后，会在itrustee_client/目录下生成dist目录，存放生成的可执行文件和动态库。





步骤5 动态库部署。

```
cp -rf dist/*.so /usr/lib64
ldconfig
```

----结束

获取 iTrustee SDK 开发套件

步骤1 获取iTrustee SDK开发套件。

下载地址： **https://gitee.com/openeuler/itrustee_sdk**

步骤2 解压iTrustee SDK开发套件，目录结构如下图所示。

```
[lhh@localhost tmp]$ cd itrustee_sdk/
[lhh@localhost itrustee_sdk]$ ll
total 40K
drwxr-xr-x. 5 lhh lhh 4.0K Nov  9 17:37 build
-rw-r--r--. 1 lhh lhh  153 Nov  9 17:37 CHANGELOG
drwxr-xr-x. 4 lhh lhh 4.0K Nov  9 17:37 include
drwxr-xr-x. 2 lhh lhh 4.0K Nov  9 17:37 License
-rw-r--r--. 1 lhh lhh  692 Nov  9 17:37 Makefile
-rw-r--r--. 1 lhh lhh  819 Nov  9 17:37 README.en.md
-rw-r--r--. 1 lhh lhh  819 Nov  9 17:37 README.md
drwxr-xr-x. 3 lhh lhh 4.0K Nov  9 17:37 src
drwxr-xr-x. 4 lhh lhh 4.0K Nov  9 17:37 test
drwxr-xr-x. 3 lhh lhh 4.0K Nov  9 17:37 thirdparty
```

----结束

说明

SDK中主要涉及“build”、“include”、“test”、“thirdparty”四个目录。其中：

- “build/”存放TA应用的签名工具。
- “include/”存放当前iTrustee OS支持的函数接口说明，遵循GP（Global Platform）标准接口协议。
- “test/”主要存放TA/CA应用源码。
- “thirdparty/”存放TA/CA应用使用的第三方库。

编译 rsa-demo 应用

步骤1 获取 rsa-demo 示例代码。

rsa-demo 代码解压后，层级目录如下：

```
[lhh@localhost tmp]$ tree -L 3 rsa-demo/
rsa-demo/
└── rsa-demo
    ├── CA
    │   └── rsa-demo
    └── TA
        └── rsa_demo
```

其中包含了TA/CA两部分代码，CA 部分需要拷贝至itrustee_sdk/test/CA目录下：

```
cp -rf rsa-demo/rsa-demo/CA/rsa-demo/ itrustee_sdk/test/CA/
```

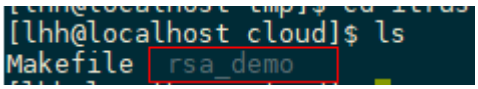
TA 部分需要拷贝至itrustee_sdk/test/TA目录下:

```
cp -rf rsa-demo/rsa-demo/TA/rsa_demo/ itrustee_sdk/test/TA/
```

步骤2 编译rsa-demo CA应用。

```
cd itrustee_sdk/test/CA/rsa-demo/cloud
make
```

编译后，会在Makefile文件同级目录产生CA二进制文件，如图下图所示。



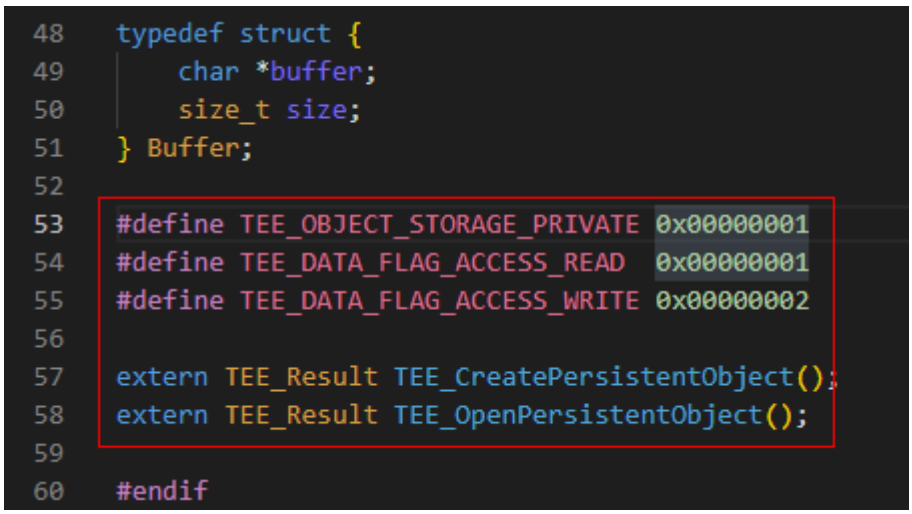
步骤3 编译rsa-demo TA应用。

修改 itrsutee_sdk/test/TA/rsa_demo/rsa_ta_interface.h 头文件，添加如下内容：

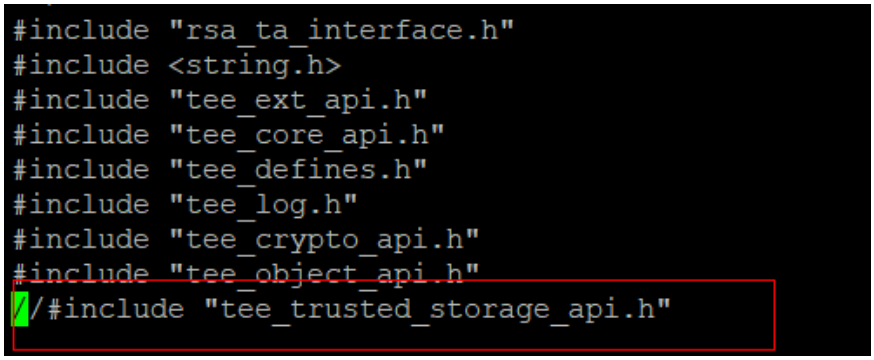
```
#define TEE_OBJECT_STORAGE_PRIVATE 0x00000001
#define TEE_DATA_FLAG_ACCESS_READ  0x00000001
#define TEE_DATA_FLAG_ACCESS_WRITE 0x00000002

extern TEE_Result TEE_CreatePersistentObject();
extern TEE_Result TEE_OpenPersistentObject();
```

添加位置如图所示:



修改 itrustee_sdk/test/TA/rsa_demo/rsa_ta_interface.c 文件，删除对"tee_trusted_storage_api.h"引用：



编译TA应用

```
cd itrustee_sdk/test/TA/rsa_demo/cloud
make
```

编译后，会在Makefile文件同级目录产生TA应用.sec文件，文件名为每个TA应用独有的UUID。

```
sign_alg value is 0x10004096
uuid str f68fd704-6eb1-4d14-b218-722850eb3ef0
=========================SUCCESS==========================
generate TA(V3 format) load image success:
/home/lhh/doc/iTrustee_Cloud_SDK/test/TA/rsa_demo/cloud/f68fd704-6eb1-4d14-b218-722850eb3ef0.sec
==========================================================
[root@localhost cloud]# ll
总用量 120K
-rw-------. 1 root root 1.1K 12月 31 14:56 config_cloud.ini
-rw-------. 1 root root  69K 12月 31 14:57 f68fd704-6eb1-4d14-b218-722850eb3ef0.sec
-rwx------. 1 root root  66K 12月 31 14:57 libcombine.so
-rw-------. 1 root root  585 12月 31 14:54 Makefile
-rw-------. 1 root root  233 12月 31 14:57 manifest.txt
drwx------. 2 root root 4.0K 12月 31 14:52 signed_config
drwx------. 2 root root 4.0K 12月 31 14:52 TA_cert
```

----结束

一个标准的TA应用，除了源代码，还应当包括TA证书、签名后的config、config_cloud.ini配置文件、manifest.txt，如下图所示。其中config_cloud.ini、config、private_key.pem与TA应用签名相关，相关文件产生方式请参考**A 调测环境TA应用开发者证书申请**。manifest.txt文件描述了TA应用在TEE侧可使用资源情况，由开发者自行提供。

```
[root@localhost TA]# tree rsa_demo/
rsa_demo/
├── cloud
│   ├── config_cloud.ini
│   ├── Makefile
│   ├── signed_config
│   │   └── config
│   └── TA_cert
│       └── private_key.pem
├── config.mk
├── manifest.txt
├── rsa_ta_interface.c
└── rsa_ta_interface.h
```

manifest.txt文件示例如下。

```
gpd.ta.appID:                f68fd704-6eb1-4d14-b218-722850eb3ef0
gpd.ta.service_name:           rsa-demo
gpd.ta.singleInstance:       true
gpd.ta.multiSession:         false
gpd.ta.instanceKeepAlive:      False
gpd.ta.dataSize:             819200
gpd.ta.stackSize:            40960
```

其中：

gpd.ta.appID类型为UUID，该UUID为用户自己生成，区别其他TA应用，并与对应的CA中的UUID保持相同。gpd.ta.service_name类型为String，表示TA应用名称，最长不超过64字符。

更多manifest.txt支持字段请参见《iTrustee SDK开发者手册》。

说明

《iTrustee SDK开发者手册》可直接向华为业务负责人获取。

3 TA/CA 应用运行环境搭建

3.1 环境要求

硬件环境要求

硬件环境要求如**表3-1**所示。

表 3-1 硬件环境

项目	版本
服务器	TaiShan 200服务器（型号2280），仅限双路服务器
主板	鲲鹏主板
BMC	1711单板（型号BC82SMMAB）
CPU	鲲鹏920处理器（型号7260、5250、5220）
机箱	不限，建议8盘或12盘

要求运行的泰山服务器已经预置了TrustZone特性，即预装iTrustee 安全OS以及配套的BMC、BIOS固件。

获取软件包

相关软件包获取方式如**表3-2**所示。

表 3-2 软件包获取

软件包名称	软件包说明	获取方法
itrustee_tzdriver	iTrustee REE侧patch包源码	**https://gitee.com/openeuler/itrustee_tzdriver**
itrustee_client	iTrustee REE侧patch包源码	**https://gitee.com/openeuler/itrustee_client**
libboundscheck	华为安全函数库	**https://gitee.com/openeuler/libboundscheck**
BoostKit-teeos_1.1.0.zip	iTrustee hpm固件包	华为技术支持网站 • 方式一：华为企业网网站下载 **BoostKit-teeos_1.1.0.zip** • 方式二：华为运营商网站下载 **BoostKit-teeos_1.1.0.zip** **说明** • 当前版本软件包受限商用，用户获取时需要提交申请且审批通过后才能下载使用。 • 集成了TrustZone特性的TaiShan服务器已经预置部署该固件，若后续有新版本iTrustee固件发布，可参考**C固件升级**进行自行升级。 下载软件包后需要校验软件包，确保与网站上的原始软件包一致。 校验方法：获取软件数字证书和软件。 在如下链接中获取校验工具和校验方法： **https://support.huawei.com/enterprise/zh/tool/pgp-verify-TL1000000054** 参见上述链接下载的《OpenPGP签名验证指南》进行软件包完整性检查

3.2 搭建步骤

CA应用需要REE（Rich Execution Environment）侧patch才能实现与TEE（Trusted Execution Environment）侧的TA应用通信，本文以OpenEuler 20.03 LTS-SP1 为例介绍REE侧patch的编译、部署。

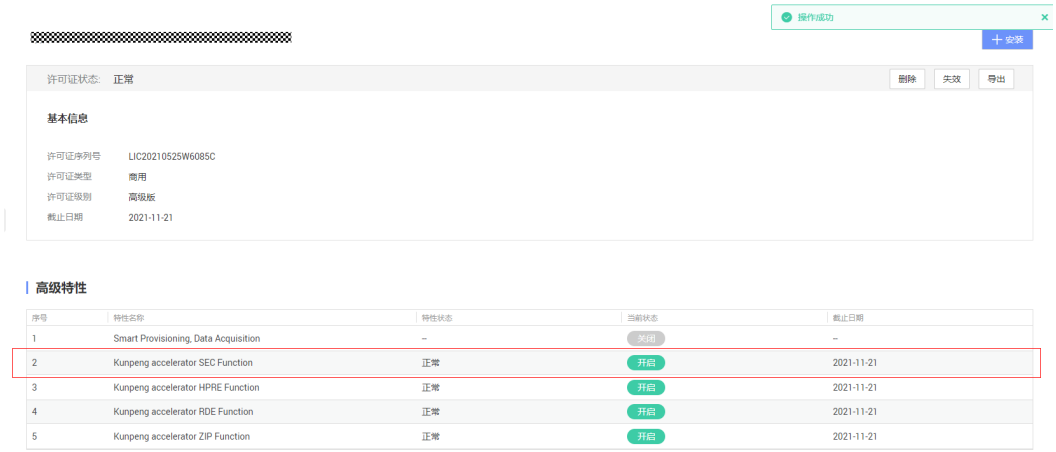
SEC 驱动 License 安装

请联系华为一线业务负责人申请泰山服务器许可证，型号iBMCV2-02-KAE-01。拿取许可证后，请完成服务器许可证安装，参考以下步骤。

步骤1 登录iBMC，依次单击“iBMC管理->许可证管理”。

步骤2 点击安装，上传申请的KAE许可证。

安装完毕后，iBMC显示“SEC特性”处于开启状态。

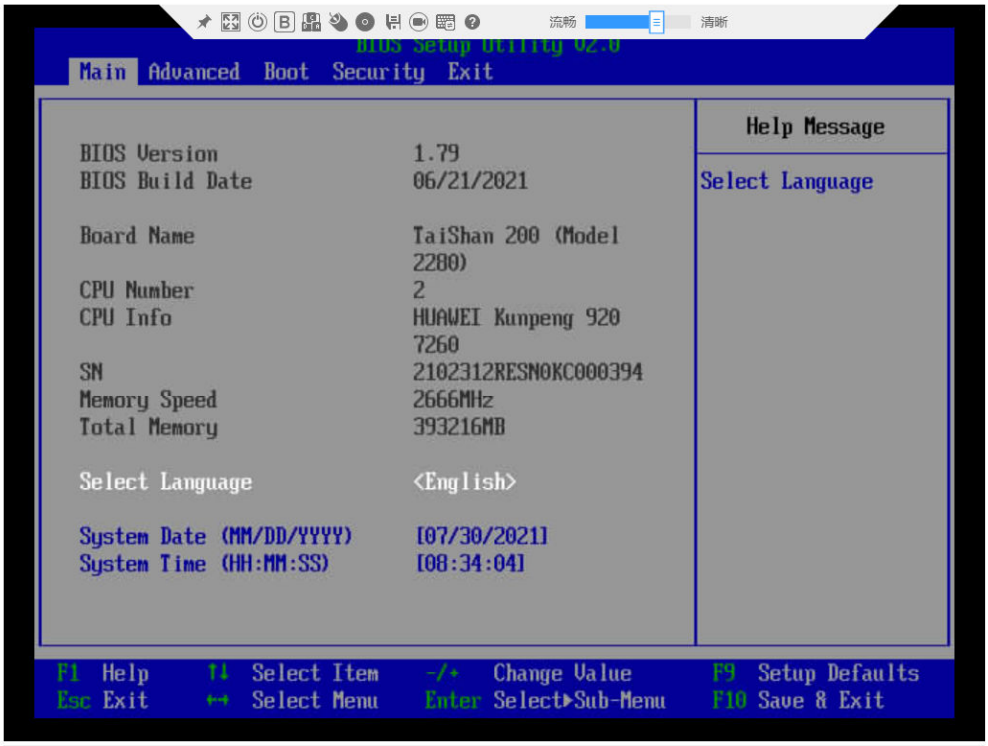


----结束

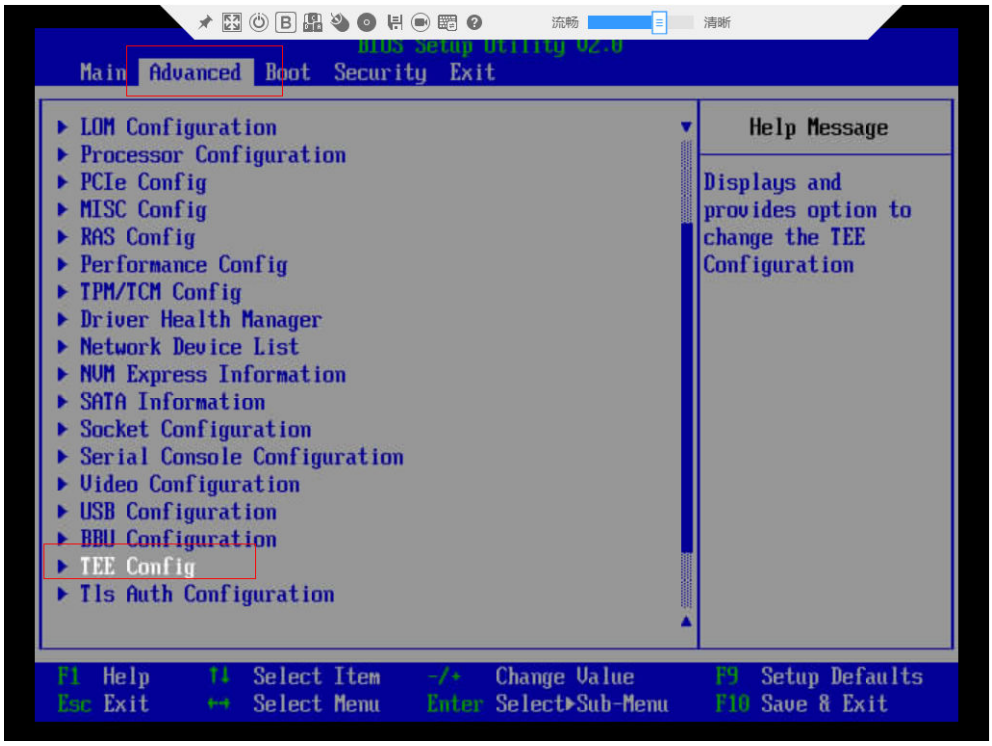
BIOS 设置

重启服务器，进入BIOS打开TrustZone特性开关，并配置TEE侧安全内存大小。

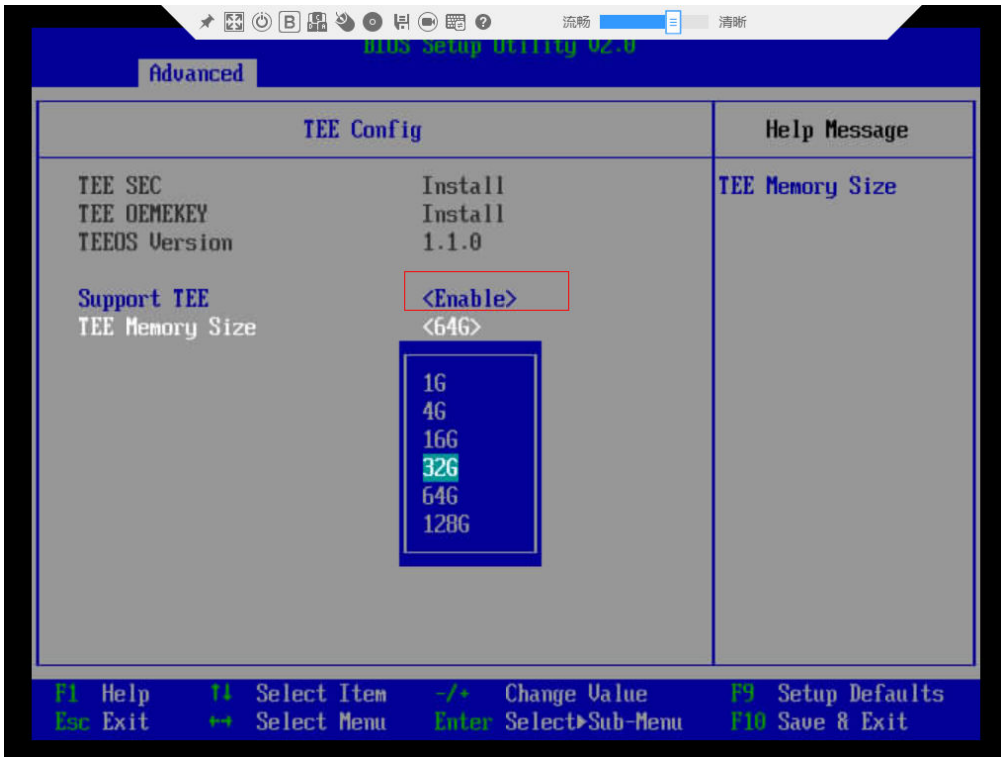
步骤1 重启服务器，进入BIOS。



步骤2 依次单击“Advanced->TEE Config”。



步骤3 打开TEE开关，根据需要配置TEE侧安全内存大小。



说明

安全内存大小选取请参考**B 安全内存规格说明**章节。

步骤4 保存BIOS配置，重启服务器，进入REE侧普通系统。

----结束

获取 iTrustee patch 包

步骤1 获取itrustee_client源码。

下载地址：**https://gitee.com/openeuler/itrustee_client**

步骤2 获取itrustee_tzdriver源码。

下载地址：**https://gitee.com/openeuler/itrustee_tzdriver**

步骤3 获取libboundscheck源码。

下载地址：**https://gitee.com/openeuler/libboundscheck**

步骤4 将libboundscheck分别放置在“itrustee_client/”目录和“itrustee_tzdriver/”目录下，并修改文件夹名称为libboundscheck，层级目录如下所示。

```
[lhh@localhost tmp]$ tree -L 1 itrustee_client/
itrustee_client/
├── include
├── libboundscheck
├── License
├── Makefile
├── README.en.md
├── README.md
└── src
```

```
root@work-PC:~# tree -L 1 itrustee_tzdriver/
itrustee_tzdriver/
├── auth
├── core
├── ko_adapt.c
├── ko_adapt.h
├── kthread_affinity
├── libboundscheck
├── License
├── Makefile
├── README.en.md
├── README.md
├── tc_ns_client.h
├── tc_ns_log.h
├── teek_client_api.h
├── teek_client_constants.h
├── teek_client_id.h
├── teek_client_type.h
├── teek_ns_client.h
└── tlogger
```

步骤5 安装编译依赖。

```
yum install openssl-devel zlib-devel
```

步骤6 编译itrustee_tzdriver。

```
cd itrustee_tzdriver && make
```

编译完成后，会生成tzdriver.ko内核模块。

```
[lhh@localhost tzdriver]$ ll
total 9.6M
drwxr-xr-x. 2 lhh lhh 4.0K Sep 23 14:46 auth
drwxr-xr-x. 2 lhh lhh 4.0K Sep 23 14:46 core
-rw-r--r--. 1 lhh lhh 5.4K Sep 18 17:16 ko_adapt.c
-rw-r--r--. 1 lhh lhh 2.9K Sep 18 17:16 ko_adapt.h
-rw-r--r--. 1 lhh lhh 228K Sep 23 14:46 ko_adapt.o
drwxr-xr-x. 2 lhh lhh 4.0K Sep 18 17:16 kthread_affinity
drwxr-xr-x. 4 lhh lhh 4.0K Sep 22 10:25 libboundscheck
drwxr-xr-x. 2 lhh lhh 4.0K Sep 18 17:16 License
-rw-r--r--. 1 lhh lhh 2.1K Sep 18 17:16 Makefile
-rw-r--r--. 1 lhh lhh   43 Sep 23 14:46 modules.order
-rw-r--r--. 1 lhh lhh   92 Sep 23 14:46 Module.symvers
-rw-r--r--. 1 lhh lhh  791 Sep 18 17:16 README
-rw-r--r--. 1 lhh lhh 4.7K Sep 18 17:16 tc_ns_client.h
-rw-r--r--. 1 lhh lhh 1.8K Sep 18 17:16 tc_ns_log.h
-rw-r--r--. 1 lhh lhh 4.8K Sep 18 17:16 teek_client_api.h
-rw-r--r--. 1 lhh lhh 4.8K Sep 18 17:16 teek_client_constants.h
-rw-r--r--. 1 lhh lhh 2.0K Sep 18 17:16 teek_client_id.h
-rw-r--r--. 1 lhh lhh 3.9K Sep 18 17:16 teek_client_type.h
-rw-r--r--. 1 lhh lhh 7.8K Sep 18 17:16 teek_ns_client.h
drwxr-xr-x. 2 lhh lhh 4.0K Sep 23 14:46 tlogger
-rw-r--r--. 1 lhh lhh 4.6M Sep 23 14:46 tzdriver.ko
-rw-r--r--. 1 lhh lhh 5.4K Sep 23 14:46 tzdriver.mod.c
-rw-r--r--. 1 lhh lhh  95K Sep 23 14:46 tzdriver.mod.o
-rw-r--r--. 1 lhh lhh 4.5M Sep 23 14:46 tzdriver.o
```

步骤7 编译itrustee_client。

```
cd itrustee_client && make
```

编译完成后，会生成dist目录，存放生成的可执行二进制和动态库。


```
[lhh@localhost libteec_vendor]$ ll
total 32K
drwxr-xr-x. 2 lhh lhh 4.0K Sep 23 14:48 dist
drwxr-xr-x. 3 lhh lhh 4.0K Sep 18 17:16 include
drwxr-xr-x. 4 lhh lhh 4.0K Sep 23 14:48 libboundscheck
drwxr-xr-x. 2 lhh lhh 4.0K Sep 18 17:16 License
-rw-r--r--. 1 lhh lhh 3.5K Sep 18 17:16 Makefile
-rw-r--r--. 1 lhh lhh 2.0K Sep 18 17:16 README.en.md
-rw-r--r--. 1 lhh lhh 2.3K Sep 18 17:16 README.md
drwxr-xr-x. 7 lhh lhh 4.0K Sep 18 17:16 src
```

```
[lhh@localhost libteec_vendor]$ ll dist
total 208K
-rwxr-xr-x. 1 lhh lhh 131K Sep 23 14:48 libboundscheck.so
-rwxr-xr-x. 1 lhh lhh  67K Sep 23 14:48 libteec.so
-rwxr-xr-x. 1 lhh lhh  67K Sep 23 14:48 teecd
-rwxr-xr-x. 1 lhh lhh  67K Sep 23 14:48 tlogcat
```

步骤8 部署itrustee_client。

```
cp -rf dist/*.so /usr/lib64 && ldconfig
cp -rf dist/teecd /usr/bin
cp -rf dist/tlogcat /usr/bin
```

----结束

说明

libboundscheck.so和libteec.so为patch驱动依赖库，需放在“/usr/lib64”目录。tlogcat提供REE侧查看TEE侧日志输出的能力，teecd为REE侧用户态守护进程，tlogcat、teecd需要放在“/usr/bin”指定目录。

加载 REE 侧驱动

步骤1 加载tzdriver.ko内核模块。

```
cd itrustee_tzdriver/
insmod tzdriver.ko && lsmod | grep tzdriver
```

```
[root@localhost patch]# lsmod | grep tzdriver
tzdriver              327680  6
```

步骤2 加载teecd守护进程。

```
/usr/bin/teecd &
ps aux | grep teecd
```

```
[root@localhost patch]# ps aux | grep teecd
root       11210  0.0  0.0 101824  1536 pts/0    Sl   15:49   0:00 /usr/bin/teecd
root       11695  0.0  0.0 214080  1600 pts/0    S+   15:51   0:00 grep --color=auto teecd
```

说明

teecd必须以绝对路径运行，即“/usr/bin/teecd”。“&”符号表示后台执行。

步骤3 查看TEE侧日志输出，确认REE侧已具备与TEE侧通信能力。

```
tlogcat
```



```
cpu0/00000:           [HM] kernel : wxn protection has enabled
cpu117/00000:         [HM] boot slave cpu:
cpu117/00000:         [HM] Logical core 117 inited
cpu117/00000:         [HM] core 75 init done
cpu0/00000:           [HM] core 76 entering...
cpu0/00000:           [HM] kernel : wxn protection has enabled
cpu118/00000:         [HM] boot slave cpu:
cpu118/00000:         [HM] Logical core 118 inited
cpu118/00000:         [HM] core 76 init done
cpu0/00000:           [HM] core 77 entering...
cpu0/00000:           [HM] kernel : wxn protection has enabled
cpu119/00000:         [HM] boot slave cpu:
cpu119/00000:         [HM] Logical core 119 inited
cpu119/00000:         [HM] core 77 init done
cpu0/00000:           [HM] core 78 entering...
cpu0/00000:           [HM] kernel : wxn protection has enabled
cpu120/00000:         [HM] boot slave cpu:
cpu120/00000:         [HM] Logical core 120 inited
cpu120/00000:         [HM] core 78 init done
cpu0/00000:           [HM] core 79 entering...
cpu0/00000:           [HM] kernel : wxn protection has enabled
cpu121/00000:         [HM] boot slave cpu:
cpu121/00000:         [HM] Logical core 121 inited
cpu121/00000:         [HM] core 79 init done
cpu0/00000:           [HM] core 7a entering...
cpu0/00000:           [HM] kernel : wxn protection has enabled
cpu122/00000:         [HM] boot slave cpu:
cpu122/00000:         [HM] Logical core 122 inited
cpu122/00000:         [HM] core 7a init done
cpu0/00000:           [HM] core 7b entering...
cpu0/00000:           [HM] kernel : wxn protection has enabled
cpu123/00000:         [HM] boot slave cpu:
cpu123/00000:         [HM] Logical core 123 inited
cpu123/00000:         [HM] core 7b init done
cpu0/00000:           [HM] core 7c entering...
cpu0/00000:           [HM] kernel : wxn protection has enabled
cpu124/00000:         [HM] boot slave cpu:
cpu124/00000:         [HM] Logical core 124 inited
cpu124/00000:         [HM] core 7c init done
cpu0/00000:           [HM] core 7d entering...
cpu0/00000:           [HM] kernel : wxn protection has enabled
cpu125/00000:         [HM] boot slave cpu:
cpu125/00000:         [HM] Logical core 125 inited
cpu125/00000:         [HM] core 7d init done
cpu0/00000:           [HM] core 7e entering...
cpu0/00000:           [HM] kernel : wxn protection has enabled
cpu126/00000:         [HM] boot slave cpu:
cpu126/00000:         [HM] Logical core 126 inited
cpu126/00000:         [HM] core 7e init done
cpu0/00000:           [HM] core 7f entering...
cpu0/00000:           [HM] kernel : wxn protection has enabled
cpu127/00000:         [HM] boot slave cpu:
cpu127/00000:         [HM] Logical core 127 inited
cpu127/00000:         [HM] core 7f init done
cpu0/00015:           [HM] detecting inactive curthread ...: state=1, currc
cpu0/00015:           [HM] SMC init finished.
cpu9/11368: 01/01 08:05:31.172 [platdrv] [error] 188:no driver can handle s
cpu9/11368:           [HM] [ERROR][120]handle swi 0xe401 failedcpu9/11368:
```

----结束

3.3 加载 TA/CA 应用

此部分内容以**编译rsa-demo应用**生成的rsa-demo 的CA/TA应用举例说明。

步骤1 拷贝CA/TA应用至指定目录。

```
mkdir -p /vendor/bin
cp -rf rsa_demo /vendor/bin
mkdir -p /data
cp -rf *.sec /data
```

说明

具体拷贝目录由TA、CA代码指定，TA、CA编码需指定实际运行时TA、CA的存放路径。更多rsa-demo编码实现请参见《Kunpeng BoostKit 21.0.RC2 TrustZone RSA Demo用户指南》。

步骤2 运行CA应用。

```
/vendor/bin/rsa_demo
```

说明

CA必须以绝对路径运行。

```
[root@localhost patch]# /vendor/bin/rsa-demoCA
random msg is :

--------------------------------------------------
bufLen = 100

2d 8e 78 64 fd b0 f9 43 8b 5e 4d e5 ef a0 6a 79 ae 42 59 be ab 99 97 97 1b b7 d4 1d 2b 35 fd 70
27 af 67 88 b3 aa d3 ee 08 89 7b 1f c8 b3 30 56 c5 03 50 9e 84 bc 97 5f 0e e9 65 86 5d e4 ca f0
80 b3 57 cf b8 7e 07 a0 21 6a 04 55 1f f3 31 5b 96 6f 08 d3 b9 ab aa bf 60 5d fd 05 19 69 18 3d
39 9a 19 88
--------------------------------------------------
encBuf is :

--------------------------------------------------
bufLen = 256

d0 24 20 44 7e e5 d9 05 d5 e2 a2 be 85 35 30 29 0c 99 07 ea ae 35 7e e8 ba 0c c1 b7 34 a3 e7 a8
78 c9 4f 2f 75 ee 4a 4f 28 8a 1f 4b d8 a3 4d c3 9f 25 a8 7a 60 8f 7c 71 b5 01 26 24 2d d8 18 cb
03 cd 9a de 9e 63 96 c3 a8 7d f6 9c 1d 39 a7 29 45 42 d8 7d ab dd 28 d0 28 4b 46 37 65 ca de 7c
da 4c f2 4b 0c 58 b3 6e b6 ba d5 48 0f b6 9a fe cb 13 b2 c3 a3 d8 52 f3 32 11 60 0a f7 2b 39 97
4c c6 94 ad 68 04 ef c1 0d 61 7a 5b 4e b2 bf a5 28 cd 40 f4 3d 5f cd 7d 7a 0f 31 a2 3b bc 13 64
50 41 b4 e3 2d 96 dc c3 bd f8 0a 45 a3 64 17 d8 f7 39 ba f0 f5 33 34 5c ad 15 2a 0d 7a 73 21 6d
d3 9a 36 7e 21 08 72 b4 69 1d 5d f9 c1 d8 9f 19 b6 8e 83 6c 63 fd d2 4e c9 a1 ae 4d 7d 69 6c 62
20 68 d8 b3 1e ac 1b 9d cb 8c 3a 3a 93 fb 0b 20 a0 c4 b1 55 36 9c 63 b7 f3 fe 57 7c 93 8d 64 20
--------------------------------------------------
derypted_buffer is :

--------------------------------------------------
bufLen = 100

2d 8e 78 64 fd b0 f9 43 8b 5e 4d e5 ef a0 6a 79 ae 42 59 be ab 99 97 97 1b b7 d4 1d 2b 35 fd 70
27 af 67 88 b3 aa d3 ee 08 89 7b 1f c8 b3 30 56 c5 03 50 9e 84 bc 97 5f 0e e9 65 86 5d e4 ca f0
80 b3 57 cf b8 7e 07 a0 21 6a 04 55 1f f3 31 5b 96 6f 08 d3 b9 ab aa bf 60 5d fd 05 19 69 18 3d
39 9a 19 88
--------------------------------------------------
signBuf is :

--------------------------------------------------
bufLen = 256

6e 01 b7 89 5a ce 8e c6 3a f8 68 f6 69 32 dd c3 db 74 74 ed a8 30 be 71 8f bb 2e a2 b3 53 f3 e5
99 28 36 7a 1c ed 7f ae 9d 33 19 b8 c5 de 9f c9 f7 5d a2 b4 cd e0 99 9e f8 39 99 52 7c d8 9c 9c
82 b0 31 24 32 f6 25 f0 9b fd ab 6c 9d ed 23 8d d6 97 fa c8 19 84 52 60 10 15 61 85 2b df 1c f5
85 ab 9f 8a 59 ae cd a5 56 53 e9 e6 14 d0 17 b0 d2 91 4d e0 d6 83 b8 38 1e 6e 7f f2 e8 f9 ae d4
42 8a d6 b4 c6 65 14 6f d2 93 cb 4f f6 17 9a 45 5f 00 41 e3 30 b9 75 7e ab b0 87 77 67 25 0b af
c9 d6 49 5d 67 97 94 ff 5d d2 4e 59 14 d3 c8 69 f9 6b 64 29 e1 22 8e 46 ff 2f 2e 09 77 11 44 fe
8e 1c 01 e3 f9 a9 a7 b3 32 f1 08 b9 f9 9c e4 7f 0f b5 8f 39 3a fe 94 ec 06 4d 49 23 d5 37 ba bc
86 1b 74 bf 25 df f6 3c 19 80 1c 01 ec 0a 8c 20 6f e9 98 2b 45 8c 2a 79 ff eb 8e f1 3c 9e 95 e4
--------------------------------------------------
Succeed to load and excute RSA crypto,sign and verify!
```

----结束

A 调测环境 TA 应用开发者证书申请

生成 configs.xml 文件

步骤1 根据manifest.txt生成手动编辑configs.xml文件。

configs.xml文件示例：

```
<?xml version="1.0" encoding="utf-8"?>
<ConfigInfo>
  <TA_Basic_Info>
     <service_name>rsa-demo</service_name>
    <uuid>f68fd704-6eb1-4d14-b218-722850eb3ef0</uuid>
  </TA_Basic_Info>
  <TA_Manifest_Info>
    <instance_keep_alive>false</instance_keep_alive>
    <stack_size>40960</stack_size>
    <heap_size>819200</heap_size>
    <multi_command>false</multi_command>
    <multi_session>false</multi_session>
    <single_instance>true</single_instance>
  </TA_Manifest_Info>
</ConfigInfo>
```

说明

“stack_size”和“heap_size”字段分别指定了TA应用可使用栈、堆最大空间，单位字节。 当前安全OS iTrustee 支持TA应用至多128多线程并发，需要为TA应用预留 256K * 2 * 128 堆空间大小。 因此TA应用'heap_size' 字段值应不小于256K * 2 * 128，例如规划TA应用功能业务需要使用 4k 堆空间，则'heap_size' 则为 （ 256 * 2 * 128 + 4 ）* 1024 字节大小。

对于栈空间“stack_size”则无此要求。

步骤2 根据manifest.txt文件实际内容对configs.xml标签内容予以替换。

----结束

TA 开发者证书申请

步骤1 生成本地密钥对。

```
openssl genrsa -out private_key.pem 4096
```

步骤2 生成CSR请求。

```
openssl req -new -key private_key.pem -out cert_req_01.csr -subj "/C=CN/O=Huawei/OU=Huawei iTrustee Production/CN=f68fd704-6eb1-4d14-b218-722850eb3ef0_rsa-demo"
```


说明

CSR生成字段/CN的值对应manifest.txt文件中uuid和service_name，并用下划线“_”隔开，请根据实际内容予以替换。

步骤3 将证书请求文件cert_req_01.csr和configs.xml文件通过PGP加密邮件发送给华为鲲鹏业务负责人，以获取由华为颁发的TA开发者证书（.pem）、签名后的config二进制文件。

说明

发送PGP加密邮件请参考**GPG安装**。

步骤4 将**步骤1**中生成的private_key.pem私钥文件存放在特定目录。

```
mv private_key.pem iTrustee_cloud_SDK/test/TA/rsa_demo/cloud/TA_cert
```

步骤5 将**3**中获取的签名后的config二进制文件存放在特定目录。

```
mv config iTrustee_cloud_SDK/test/TA/rsa_demo/cloud/signed_config
```

步骤6 配置config_cloud.ini文件中TA私钥、config存放路径。

```
releaseType = 1
;;;
;0 means TA not installed by OTRP
;1 means TA installed by OTRP
otrpFlag = 0
;;;
;server address for signing TA
serverIp=
;;;
;public key for encrypt TA
;[fixed value]
encryptKey = cloud/rsa_public_key_cloud.pem
;;;
;public key length
;[fixed value]
encryptKeyLen = 3072
;;;
;0 means not sign
;1 means signed by local private
;2 means signed using native sign tool;
;3 means signed by CI
;[fixed value]
signType = 1
;;;
;private key for signing TA
;[private key owned by yourself]
signKey = ../../test/TA/taishan-ta/cloud/TA_cert/private_key.pem
;;;
;private key length for signing TA
;[key length should be 4096 for security enhance]
signKeyLen = 4096
;;;
;0 means SHA256 hash type
;1 means SHA512 hash type
;[set value to 0 by default]
hashType = 0
;;;
;0 means padding type is pkcs1v15
;1 means padding type is PSS
;[set value to 0 by default]
paddingType = 0
;;;
;config file
;[signed config file by Huawei]
configPath= ../../test/TA/taishan-ta/cloud/signed_config/config
```

----结束

须知

为方便ISV快速体验开发过程，以上过程仅演示了最快捷方便的操作步骤，可用于调测环境。在商用业务开发环境中，保护ISV开发者证书签名密钥极其重要，相关密钥的安全使用方法由ISV自行设计和负责。

建议的方法包括但不限于：

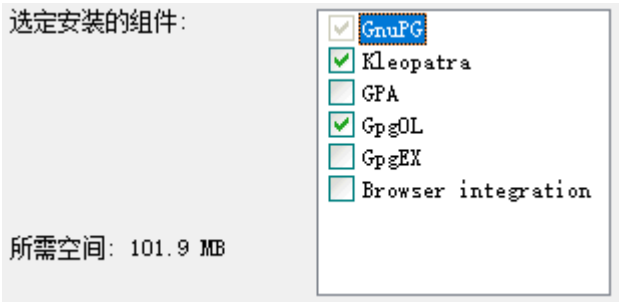
1. 以安全、防篡改、加密的方式保护签名密钥，例如使用硬件安全模块（HSM，Hardware Security Module）。
2. 物理安全，这包括限制人员物理访问签名设施/平台，以及登录私钥加密保护模块。
3. 签名过程在KMS（Key Management System）上进行，确保版本签名无人员接触。

GPG 安装

PGP是一种数据加密和解密算法，提供数据通信加密和认证，常用于签名、加密、解密文件、电子邮件等。下文介绍了在Windows平台发送PGP加密邮件的步骤。

步骤1 从官方网站下载Gpg4win软件。

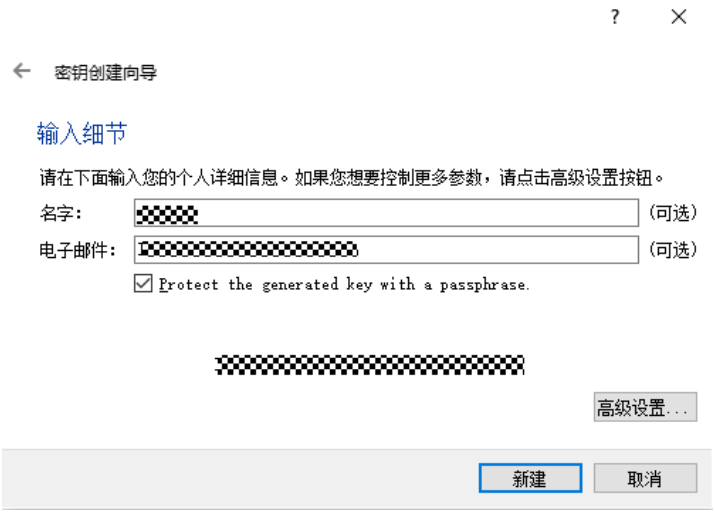
步骤2 安装Kleopatra和GpgOL。



步骤3 安装完毕后，运行Kleopatra，依次选择“文件 > 新建密钥对 > 创建个人OpenPGP密钥对”。

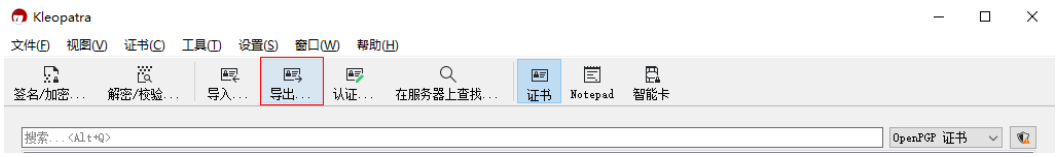
打开密钥创建向导窗口。

步骤4 输入名字、电子邮件地址和passphrase完成密钥对创建。



步骤5 运行Kleopatra，选择需要导出公钥的用户，单击“导出”，选择需要保存到的目录。

导出后的公钥文件后缀名为asc。

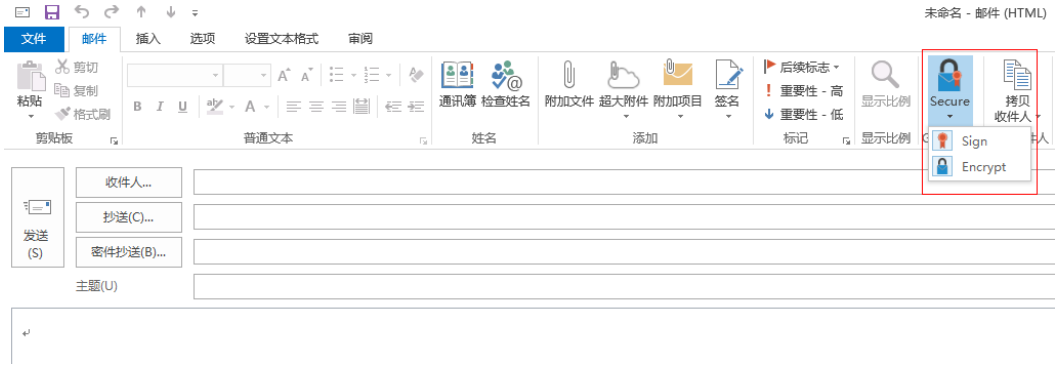


说明

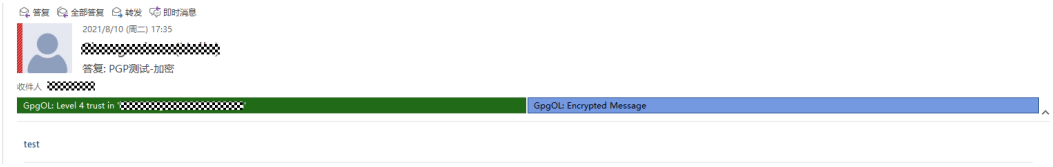
如果需要发送PGP加密邮件，需要将发件人的asc公钥导入到收件人的Kleopatra软件中；同理，发件人也需要导入收件人提供的asc公钥文件。

步骤6 打开Outlook，创建新邮件。

步骤7 在工具栏中选在GpgOL插件，勾选“Encrypt”和“Sign”，单击发送邮件。



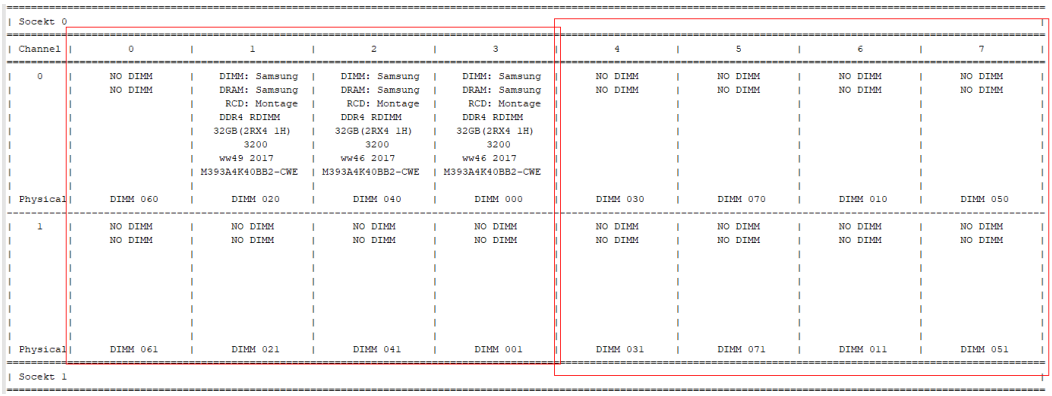
此时收件人则可以看到带有加密、签名标记的邮件。



----结束

B 安全内存规格说明

图 B-1 内存卡槽使用情况



Socekt 0								
Channel	0	1	2	3	4	5	6	7
0	NO DIMM NO DIMM	DIMM: Samsung DRAM: Samsung RCD: Montage DDR4 RDIMM 32GB(2RX4 1H) 3200 ww49 2017 M393A4K40BB2-CWE	DIMM: Samsung DRAM: Samsung RCD: Montage DDR4 RDIMM 32GB(2RX4 1H) 3200 ww46 2017 M393A4K40BB2-CWE	DIMM: Samsung DRAM: Samsung RCD: Montage DDR4 RDIMM 32GB(2RX4 1H) 3200 ww46 2017 M393A4K40BB2-CWE	NO DIMM NO DIMM	NO DIMM NO DIMM	NO DIMM NO DIMM	NO DIMM NO DIMM
Physical	DIMM 060	DIMM 020	DIMM 040	DIMM 000	DIMM 030	DIMM 070	DIMM 010	DIMM 050
1	NO DIMM NO DIMM	NO DIMM NO DIMM	NO DIMM NO DIMM	NO DIMM NO DIMM	NO DIMM NO DIMM	NO DIMM NO DIMM	NO DIMM NO DIMM	NO DIMM NO DIMM
Physical	DIMM 061	DIMM 021	DIMM 041	DIMM 001	DIMM 031	DIMM 071	DIMM 011	DIMM 051
Socekt 1								

图B-1显示了CPU0的内存卡槽使用情况。TaiShan服务器具有两个CPU，即图中Socket 0和未在图中显示的Socket 1。一个CPU拥有2个Totem Die，如图中红框标注。

其中每个Die可以插入4个通道共8条内存，默认情况下，Die内内存交织开启，Die间内存交织关闭。如需开启Die间内存交织功能，请到BIOS中手动开启。如果开启Die间交织，两个Die使用的内存通道、内存大小需保持一致。

当前版本安全内存选择具有如下约束。

1. 安全内存应小于总物理内存大小，需要预留部分内存给到普通REE侧（BIOS、操作系统应用内存）。
2. 安全内存应小于一个MSD（Memory Space Decoder）窗口大小。
3. 启用3路、6路内存交织，BIOS会申请3段同偏移大小安全内存，安全内存应小于MSD窗口1/3。
4. 鉴于Socket间交织性能影响，安全内存只能在单个CPU内存空间内分配。

不同内存插法具有不同的MSD窗口，以一个Die内举例，单条内存大小为32G，映射关系如表B-1所示。

表 B-1 MSD 窗口大小计算

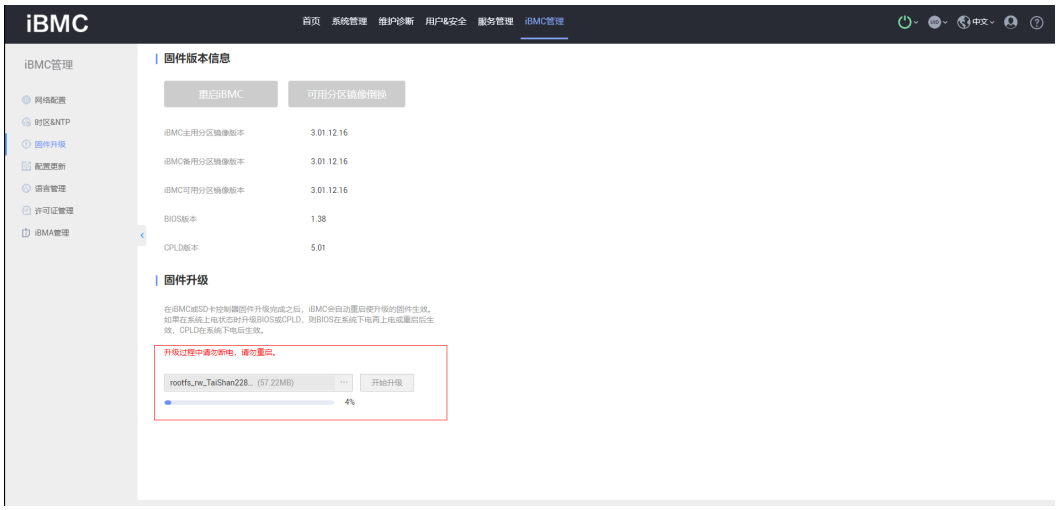
内存条数量	MSD窗口大小	说明
x	x * 32G	其中 x = 1, 2, 3, 4, 6, 8
5	2 * 32G, 3 * 32G	可理解为2路交织 + 3路交织，因此具有两个MSD窗口
7	3 * 32G, 4 * 32G	可理解为3路交织 + 4路交织，因此具有两个MSD窗口

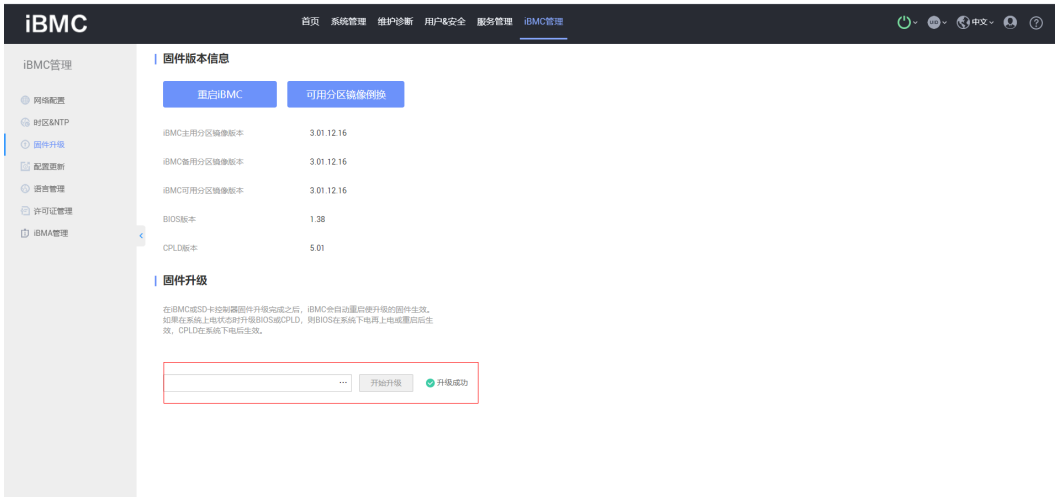
在存在多个MSD窗口时，系统优先从小的MSD窗口尝试分配安全内存，无法分配时再从下一个MSD窗口分配。应尽可能避免3路、6路内存交织（存在一定内存浪费），推荐2，4，8路内存插法。

C 固件升级

如果后续有新的固件（BMC、BIOS或安全OS）版本升级，可参考以下步骤。

步骤1 登录iBMC，依次单击“iBMC管理->固件升级”，选择BMC新版本HPM包，点击开始升级。

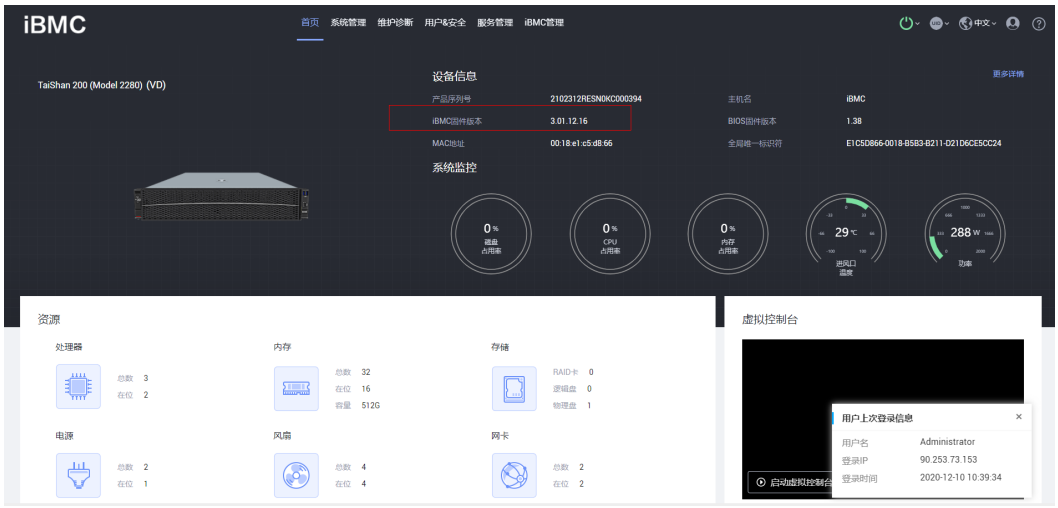




说明

Web页面提示升级成功后，BMC需要几分钟完成重启操作，在此期间无法登录iBMC。待完成重启后，可重新登入。

步骤2 BMC升级完毕后，重新登录iBMC可查看新版BMC固件版本。



----结束

D 缩略语

C		
CA	Client Application	客户端应用程序
G		
GP	Global Platform	GP标准组织
H		
HSM	Hardware Security Module	硬件安全模块
K		
KMS	Key Management System	密钥管理系统
M		
MSD	Memory Space Decoder	内交织后内存空间大小
N		
NDA	Non-Disclosure Agreement	保密协议
R		
REE	Rich Execution Environment	业务执行环境
T		
TA	Trusted Application	可信应用程序
TEE	Trusted Execution Environment	可信执行环境

E 修订记录

发布日期	修改说明
2020-01-19	第二次正式发布。 “**A 调测环境TA应用开发者证书申请**”中新增密钥的安全使用方案相关建议。
2022-01-06	第一次正式发布。