阿里云
IoT可信执行环境

开发手册

文档版本：20190802

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。
2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。
3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。
4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的”现状“、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。
5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含”阿里云”、Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。
6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	 **禁止:** 重置操作将丢失用户配置数据。
	该类警示信息可能导致系统重大变更甚至故障，或者导致人身伤害等结果。	 **警告:** 重启操作将导致业务中断，恢复业务所需时间约10分钟。
	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	 **说明:** 您也可以通过按Ctrl + A选中全部文件。
>	多级菜单递进。	设置 > 网络 > 设置网络类型
粗体	表示按键、菜单、页面名称等UI元素。	单击 确定。
`courier` 字体	命令。	执行 `cd /d C:/windows` 命令，进入Windows系统文件夹。
*## *	表示参数、变量。	`bae log list --instanceid` *`Instance_ID`*
[]或者[a|b]	表示可选项，至多选择一个。	`ipconfig` *`[-all|-t]`*
{}或者{a|b}	表示必选项，至多选择一个。	`swich` *`{stand | slave}`*

目录

1 TEE Pro试用版指南

2 Uboot/Kernel 编译指南

工具链

交叉编译工具链请使用如下推荐版本：

```
gcc-linaro-arm-linux-gnueabihf-4.9-2014.08_linux
```

说明：

下载地址

Uboot版本

uboot-imx-rel_imx_4.1.15_2.0.0_ga

下载使用说明

```
$ wget http://git.freescale.com/git/cgit.cgi/imx/uboot-imx.git/
snapshot/uboot-imxrel\_imx\_4.1.15\_2.0.0\_ga.tar.bz2
$ tar xjvf ~/uboot-imxrel\_imx\_4.1.15\_2.0.0\_ga.tar.bz2
```

Linux版本

linux-imx-rel_imx_4.1.15_2.0.0_ga

下载使用说明：

```
$ wget http://git.freescale.com/git/cgit.cgi/imx/linux-imx.git/
snapshot/linux-imx-rel_imx_4.1.15_2.0.0_ga.tar.bz2
$ tar xjvf ~/linux-imx-rel\_imx\_4.1.15\_2.0.0\_ga.tar.bz2
```

i.MX6UL 平台

link-security-tee-pro-imx6ul-patch：

```
0001-uboot-add-tee-pro-support.patch
0001-linux-add-tee-pro-support.patch
```

说明：

下载地址：TEE pro 试用版

Uboot patch

将0001-uboot-add-tee-pro-support.patch置于和uboot-imx-rel_imx_4.1.15_2.0.0_ga 同级目录下，运行命令：

```
$ patch -p1 < 0001-uboot-add-tee-pro-support.patch
$ cd uboot-imx-rel_imx_4.1.15_2.0.0_ga
$ export CROSS_COMPILE=arm-linux-gnueabihf-
$ make ARCH=arm mx6ul_14x14_evk_tee_defconfig
$ make ARCH=arm
```

编译生成文件：

```
uboot-imx-rel_imx_4.1.15_2.0.0_ga/u-boot.imx
```

Linux patch

将0001-linux-add-tee-pro-support.patch置于和linux-imx-rel_imx_4.1.15_2.0.0_ga 同级目录下，运行命令：

```
$ patch -p1 < 0001-linux-add-tee-pro-support.patch
$ cd linux-imx-rel_imx_4.1.15_2.0.0_ga
$ make ARCH=arm imx_v7_defconfig
$ make ARCH=arm zImage
$ make ARCH=arm imx6ul-14x14-evk.dtb
```

编译生成文件：

```
linux-imx-rel_imx_4.1.15_2.0.0_ga/arch/arm/boot/zImage
linux-imx-rel_imx_4.1.15_2.0.0_ga/arch/arm/boot/dts/imx6ul-14x14-evk.
dtb
```

i.MX6QP 平台

link-security-tee-pro-imx6qp-patch：

```
0001-uboot-add-tee-pro-support.patch
0002-uboot-add-6qp-tee-support.patch
0001-linux-add-tee-pro-support.patch
0002-linux-add-6qp-tee-support.patch
```

Uboot patch

将0001-uboot-add-tee-pro-support.patch，0002-uboot-add-6qp-tee-support.patch置于和 uboot-imx-rel_imx_4.1.15_2.0.0_ga 同级目录下，运行命令：

```
$ patch -p1 < 0001-uboot-add-tee-pro-support.patch
$ patch -p1 < 0002-uboot-add-6qp-tee-support.patch
$ cd uboot-imx-rel_imx_4.1.15_2.0.0_ga
$ export CROSS_COMPILE=arm-linux-gnueabihf-
$ make ARCH=arm mx6qpsabresd_config
```

```
$ make ARCH=arm
```

变异生成文件：

```
uboot-imx-rel_imx_4.1.15_2.0.0_ga/u-boot.imx
```

Linux patch

将0001-linux-add-tee-pro-support.patch, 0002-linux-add-6qp-tee-support.patch置于和linux-imx-rel_imx_4.1.15_2.0.0_ga 同级目录下，运行命令：

```
$ patch -p1 < 0001-linux-add-tee-pro-support.patch
$ patch -p1 < 0002-linux-add-6qp-tee-support.patch
$ cd linux-imx-rel_imx_4.1.15_2.0.0_ga
$ make ARCH=arm imx_v7_defconfig
$ make ARCH=arm zImage
$ make ARCH=arm imx6qp-sabresd.dtb
```

编译生成文件：

```
linux-imx-rel_imx_4.1.15_2.0.0_ga/arch/arm/boot/zImage
linux-imx-rel_imx_4.1.15_2.0.0_ga/arch/arm/boot/dts/imx6qp-sabresd.dtb
```

3 Image 烧录指南

Uboot

请参考开发板对uboot的烧录方法，将编译生成的 u-boot.imx 烧录到开发板中

Kernel

请参考开发板对uboot的烧录方法，将编译生成的 zImage 和 DTB 文件烧录到开发板中

tee.img

参考 zImage 文件的更新方法，将 tee.img 置于开发板中 zImage 同级目录

重启开发板，系统会自动启动，直到出现Linux命令提示符。或者可以在uboot的命令行中输入命令来加载执行

i.MX6UL：

```
run findfdt;run loadfdt;run loadimage;fatload mmc 1:1 0x84000000 tee.
img;run mmcargs;bootm 0x84000000 - 0x83000000
```

i.MX6QP：

```
run findfdt;run loadfdt;run loadimage;fatload mmc 1:1 0x20000000 tee.
img;run mmcargs;bootm 0x20000000 - 0x18000000
```

4 TEE Pro 启动流程

重启开发板，系统会自动启动，直到出现Linux命令提示符。 或者可以在 uboot 的命令行中输入命令来加载执行：

i.MX6UL：

```
run findfdt;run loadfdt;run loadimage;fatload mmc 1:1 0x84000000 tee.
img;run mmcargs;bootm 0x84000000 - 0x83000000
```

i.MX6QP：

```
run findfdt;run loadfdt;run loadimage;fatload mmc 1:1 0x20000000 tee.
img;run mmcargs;bootm 0x20000000 - 0x18000000
```

确认log打印

1，启动log

```
*************************************************
*     Alibaba Cloud Link Security TEE Pro
* This is a trial version!!!
* Version number: 1.0.4.
*************************************************
```

2，加载TEE驱动log

```
TZDD Version: 1.0.4
TEE Debug: buf: 0xbee99800, size: 0x66400, ctl: 0xbf099800
TEE Debug Buf: 0xbee99838, size: 0x662c8, ctl: 0xbf099800
I tzdd_proxy_thread(228) tzdd_proxy_thread: cpu id(0)
```

3，登录log

```
Freescale i.MX Release Distro 4.1.15-2.0.0 imx6qdlsolo /dev/ttymxc0

imx6qdlsolo login:
```

启动tstd守护进程

1，在开发板 linux 根目录创建 /tee 目录

2，将 tstd 程序复制到 tee 目录

3，在 tee 目录下创建 tst 目录

4，运行命令 ./tstd ./tst & 启动 tstd 守护进程

TEE Pro启动完毕

5 Sample 测试指南

以XOR程序为例

工具链

交叉编译工具链请使用如下推荐版本：

```
gcc-linaro-arm-linux-gnueabihf-4.9-2014.08_linux
```

说明:

下载地址

编译 TA（Trust Application）

将 PATH=~/gcc-linaro-arm-linux-gnueabihf-4.9-2014.08_linux/bin:$PATH 加入~/.bashrc

1，在 sample/xor/tw 目录下执行 make –f Makefile.sdk

2，当前目录生成文件 ta_xor.bin

生成新的tee.img

```
复制编译生成的 ta_xor.bin 到 $/tw/tools 目录
复制 $/tw/bin/ 目录下的 atf.bin，tee.bin 到 $/tw/tools 目录
运行命令 ./srv_tk add -b tee.bin -s ta_xor.bin -o tee.bin
运行脚本 ./rel.sh
生成最终可加载执行文件 tee.img
将生成的 tee.img 按照前文描述烧录到开发板中
```

编译 CA（Client Application）

将 PATH=~/gcc-linaro-arm-linux-gnueabihf-4.9-2014.08_linux/bin:$PATH 加入~/.bashrc，在 samples/xor/ntw 目录下执行“make –f Makefile.sdk”

当前目录生成执行的文件 ca_xor

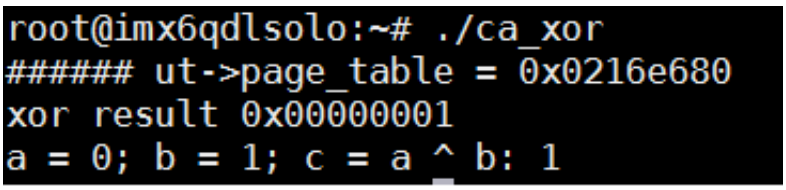
运行CA

```
1，烧录新生成的 tee.img 后重启开发板
2，将编译生成的 ca_xor 复制到先前在开发板上创建的 tee 目录中
```

3，运行 ./ca_xor

运行结果

如下图：



6 应用开发说明

CA Sample

```
/* UUID和CMD需要和TA里的定义保持一致 */
#define UUID_XOR    { 0x13245768, 0xacbd, 0xcedf,   \
                    { 0x01, 0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78
 } }

#define CMD_XOR     (0x01234567)

/* 设置UUID，对应TA，每个TA有唯一的UUID */
static const TEEC_UUID _g_uuid_xor = UUID_XOR;

int main(int argc, char *argv[])
{
    unsigned int     a = 0, b = 1, c = 0;
    TEEC_Context     context;
    TEEC_Session     session;
    TEEC_Operation   operation;
    TEEC_Result      result = TEEC_SUCCESS;

    _DEPRESS_UNUSED_WARNING(argc);
    _DEPRESS_UNUSED_WARNING(argv);

    /* Step1: 初始化Context */
    result = TEEC_InitializeContext(NULL, &context);
    if (result != TEEC_SUCCESS) {
        ERR_LOG("TEEC_InitializeContext", result);
        goto cleanup1;
    }

    /* Step2: 打开对应TA的Session */
    result = TEEC_OpenSession(&context, &session,
            &_g_uuid_xor, TEEC_LOGIN_USER, NULL, NULL, NULL);
    if (result != TEEC_SUCCESS) {
        ERR_LOG("TEEC_OpenSession", result);
        goto cleanup2;
    }

  /* 初始化对应的参数，如下2个为输入，1个为输出 */
    operation.paramTypes = TEEC_PARAM_TYPES(
            TEEC_VALUE_INPUT, TEEC_VALUE_INPUT,
            TEEC_VALUE_OUTPUT, TEEC_NONE);
    operation.params[0].value.a = a;
    operation.params[1].value.a = b;
    operation.params[2].value.a = c;

  /* Step3: 开始具体的命令调用 */
    result = TEEC_InvokeCommand(&session, CMD_XOR, &operation, NULL);
    if (result != TEEC_SUCCESS) {
        ERR_LOG("TEEC_InvokeCommand", result);
        goto cleanup3;
    }

    c = operation.params[2].value.a;

    printf("a = %d; b = %d; c = a ^ b: %d\n", a, b, c);
```

```
  cleanup3:
    /* Step4: 关闭Session */
      TEEC_CloseSession(&session);
  cleanup2:
    /* Step5: 释放Context */
      TEEC_FinalizeContext(&context);
  cleanup1:
      return result;
}
```

TA Sample

```
/* UUID和CMD需要和CA里的定义保持一致 */
#define UUID_XOR    { 0x13245768, 0xacbd, 0xcedf,   \
                    { 0x01, 0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78
 } }

#define CMD_XOR     (0x01234567)

static TEE_UUID xor_uuid = UUID_XOR;

static TEE_Result _TA_CreateEntryPoint(void)
{
    return TEE_SUCCESS;
}

static void _TA_DestroyEntryPoint(void)
{
    return;
}

static TEE_Result _TA_OpenSessionEntryPoint(
        uint32_t paramTypes, TEE_Param params[4], void **sessionCon
text)
{
  /* CA调用OpenSession函数时的对应操作可加入其中 */
    return TEE_SUCCESS;
}

static void _TA_CloseSessionEntryPoint(void *sessionContext)
{
    return;
}

/* CA调用InvokeCommand函数时的对应操作可加入其中 */
static TEE_Result _TA_InvokeCommandEntryPoint(
        void *sessionContext, uint32_t commandID,
        uint32_t paramTypes, TEE_Param params[4])
{
    if (TEE_PARAM_TYPES(
                TEE_PARAM_TYPE_VALUE_INPUT,
                TEE_PARAM_TYPE_VALUE_INPUT,
                TEE_PARAM_TYPE_VALUE_OUTPUT,
                TEE_PARAM_TYPE_NONE) != paramTypes) {
        return TEE_ERROR_BAD_PARAMETERS;
    }

    if (CMD_XOR != commandID) {
        return TEE_ERROR_BAD_PARAMETERS;
    }
```

```
    params[2].value.a = params[0].value.a ^ params[1].value.a;

    TEE_Print("xor result 0x%08x\n", params[2].value.a);

    return TEE_SUCCESS;
}

TEE_SRV_INFO_START(
        _TA_CreateEntryPoint,
        _TA_DestroyEntryPoint,
        _TA_OpenSessionEntryPoint,
        _TA_CloseSessionEntryPoint,
        _TA_InvokeCommandEntryPoint)
TEE_SRV_PROP_UUID("gpd.ta.appID", UUID_XOR)
TEE_SRV_PROP_BOOL("gpd.ta.singleInstance", false)
TEE_SRV_PROP_BOOL("gpd.ta.multiSession", true)
TEE_SRV_PROP_BOOL("gpd.ta.instanceKeepAlive", false)
TEE_SRV_PROP_INT("gpd.ta.dataSize", 0x1000)
TEE_SRV_PROP_INT("gpd.ta.stackSize", 0x800)
TEE_SRV_INFO_END
};
```

7 错误代码

Name	Value	Description / Cause
TEEC_SUCCESS	0x00000000	The operation was successful.
TEEC_ERROR_GENERIC	0xFFFF0000	Non-specific cause.
TEEC_ERROR_ACCESS_DENIED	0xFFFF0001	Access privileges are not sufficient.
TEEC_ERROR_CANCEL	0xFFFF0002	The operation was cancelled.
TEEC_ERROR_ACCESS_CONFLICT	0xFFFF0003	Concurrent accesses caused conflict.
TEEC_ERROR_EXCESS_DATA	0xFFFF0004	Too much data for the requested operation was passed.
TEEC_ERROR_BAD_FORMAT	0xFFFF0005	Input data was of invalid format.
TEEC_ERROR_BAD_PARAMETERS	0xFFFF0006	Input parameters were invalid.
TEEC_ERROR_BAD_STATE	0xFFFF0007	Operation is not valid in the current state.
TEEC_ERROR_ITEM_NOT_FOUND	0xFFFF0008	The requested data item is not found.
TEEC_ERROR_NOT_IMPLEMENTED	0xFFFF0009	The requested operation should exist but is not yet implemented.
TEEC_ERROR_NOT_SUPPORTED	0xFFFF000A	The requested operation is valid but is not supported in this Implementation.
TEEC_ERROR_NO_DATA	0xFFFF000B	Expected data was missing.
TEEC_ERROR_OUT_OF_MEMORY	0xFFFF000C	System ran out of resources.
TEEC_ERROR_BUSY	0xFFFF000D	The system is busy working on something else.
TEEC_ERROR_COMMUNICATION	0xFFFF000E	Communication with a remote party failed.
TEEC_ERROR_SECURITY	0xFFFF000F	A security fault was detected.
TEEC_ERROR_SHORT_BUFFER	0xFFFF0010	The supplied buffer is too short for the generated output.