运营技术广角

基于 TEE 的移动终端数据安全研究与实现

陈淑珍,杨敏维,何峣

(中国电信股份有限公司广州研究院,广东 广州 510630)

摘　要:分析对比了移动终端数据安全的 3 种实现方式,提出了基于 TEE 的移动终端数据安全框架,并探讨了典型 TEE 移动支付方案。在典型方案基础上,创新性地提出了基于 TEE 的指纹动态调整支付额度的业务方案,最后给出了核心代码实现。

关键词:移动终端;数据安全;移动支付;TEE

中图分类号:TN929.5　　**文献标识码**:A

doi: 10.11959/j.issn.1000-0801.2017036

Research and implementation of data security for mobile terminal based on TEE

CHEN Shuzhen, YANG Minwei, HE Yao

Guangzhou Research Institute of China Telecom Co., Ltd., Guangzhou 510630, China

Abstract: The three kinds of data security scheme for mobile terminal were analyzed and compared. The framework of data security for mobile terminal based on TEE was proposed, and the typical mobile payment scheme based on TEE was discussed. On the basis of the typical scheme, an innovative business plan based on TEE to adjust the payment amount of fingerprintsdynamically was proposed, and finally the core code was given.

Key words: mobile terminal, data security, mobile payment, trusted execution environment

1　引言

随着移动互联网及移动支付的普及,移动终端数据安全日益重要。由全球平台组织(Global Plantform,GP)制定并推进的可信执行环境 (trusted execution environment,TEE)标准,是面向移动终端,由硬件实现,通过开辟移动终端主处理器内部的安全区域,提供一个隔离的可信执行环境。终端通过分离 TEE 和 REE (rich execution environment,富执行环境)的软硬件资源,实现对敏感数据的存储与保护,确保 TEE 内代码和数据的安全性、机密性以及完整性。TEE 的安全级别比 REE 安全级别更高,能够满足大多数应用的安全需求。

特别在移动支付、移动办公等对安全需求较高的领域,将指纹识别+TEE 等技术引入移动终端,是一种更安全的数据安全保护机制,也成为产业链众多终端厂商的选择。

2　移动终端数据安全实现方式

目前,产业链实现移动终端数据安全的方式主要有如下 3 种,如图 1 所示。

(1)单域应用级

单域应用级的安全机制就是 permission 机制。应用访问系统敏感数据或者特权资源, 必须在 AndroidManifest.xml 配置文件中进行权限申请,并在安装时由用户决定是否赋予相应的权限。单域应用级数据安全分为两种方式:

收稿日期:2016-12-14;修回日期:2017-01-23

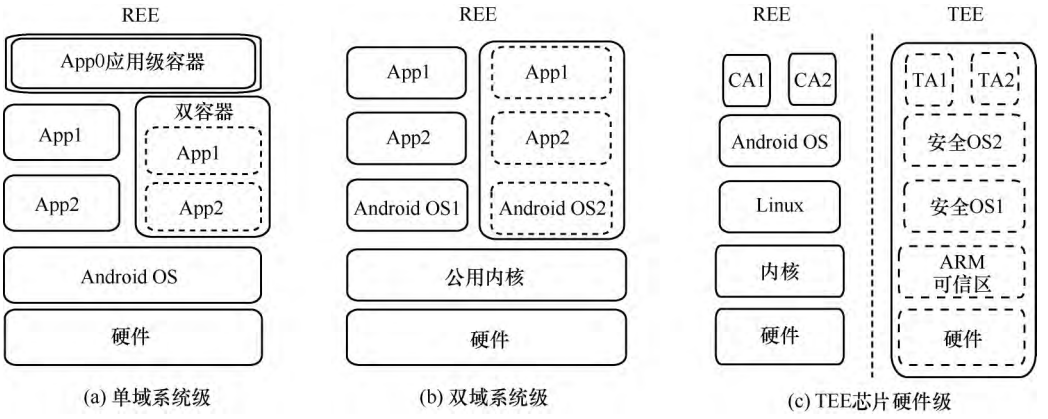


图 1　移动终端数据安全 3 种实现方式

应用级容器和双容器。应用级容器采用完全的应用层数据隔离，提供基础加解密功能，如沙箱技术、权限管控、SEAndroid 等；双容器基于系统层的双存储空间，实现数据软隔离，如 LXC、Docker Cgroups 技术等。市面上终端厂商实采用单域应用级方式实现“隐私空间”，完成数据加解密和隔离存储，如三星 GalaxyS4/5 的 KNOX 应用。

(2)双域系统级

基于系统层的双存储空间，实现数据软隔离。双域系统级数据安全分为两种方式：操作系统虚拟化和双操作系统。操作系统虚拟化是同一个操作系统，为不同应用创建隔离的、独立的执行环境，并赋予不同的权限。双操作系统是同一个公共内核、REE 内，分别安装两个不同的操作系统，实现数据软隔离。市面上主流安全终端，均基于双域系统构建了安全办公环境、安全支付环境，如酷派铂顿。

(3)TEE 芯片硬件级

基于芯片的双操作系统，实现数据硬隔离。TEE 芯片硬件级与双域系统级的最大区别是：TEE 是硬件实现，集成在移动终端处理器内部，硬件隔离出两个执行环境：REE 和 TEE，是独立的两个安全体系。REE 是普通的非保密执行环境，运行普通终端操作系统，如 Android；TEE 是安全的保密执行环境，硬件部分以硬件隔离提供基础安全硬件能力(如安全存储、安全加密、安全输入输出)，目前芯片厂商主要采用 ARM 可信区，技术成熟；软件部分需加载安全操作系统，如 MobiCore、SEAndroid 等，并基于安全操作系统与上层应用、REE、SE 交互，提供密钥存储、密钥加解密、密钥验证、敏感数据存储、可信应用等系列安全服务。三星 Galaxy S3 欧版中，普通域安装 Android，安全域(security domain, SD)采用 MobiCore 安全 OS，构建安全的数据版权管理。华为 Mate7 基于 TEE 安装 SEAndroid，进行指纹安全存储等。

移动终端 3 种数据安全实现方式对比见表 1。

3　基于 TEE 的移动终端数据安全框架

由于 TEE 芯片硬件级可实现数据硬隔离，为数据安全提供更高的安全性，成为了主流数据安全方式。如图 2 所示，基于 TEE 的移动终端数据安全框架主要分为两部

表 1　移动终端数据安全 3 种实现方式对比

	单域应用级	双域系统级	TEE 芯片硬件级
密钥存储	应用级存储	REE 安全操作系统存储	TEE 存储
敏感数据加解密	应用级加密	REE 安全操作系统执行	TEE 执行
应用下载	无验证或应用级验证(如 360 杀毒等)	无验证或应用级验证(如 360 杀毒等)	TEE 和管理系统的授权下载、下载前会验证和认证、下载过程中会校验
应用访问	应用控制	应用和用户控制	TEE 控制 TA 应用安全访问
隔离	应用级隔离，如沙箱模式、权限管控等	系统级隔离，如多用户、双系统、虚拟化等	硬件实现 REE 和 TEE 隔离
防软件攻击能力	未经认证的 OS 保护； 有限 boot 完整性检查	未经认证的 OS 保护； 有限 boot 完整性检查	对富操作系统和外设的安全访问由经认证的 TEE OS 保证
防硬件攻击能力	基本没有保护	基本没有保护	依赖于 TEE 实现及宿主平台的硬件特性，包括安全应用部署

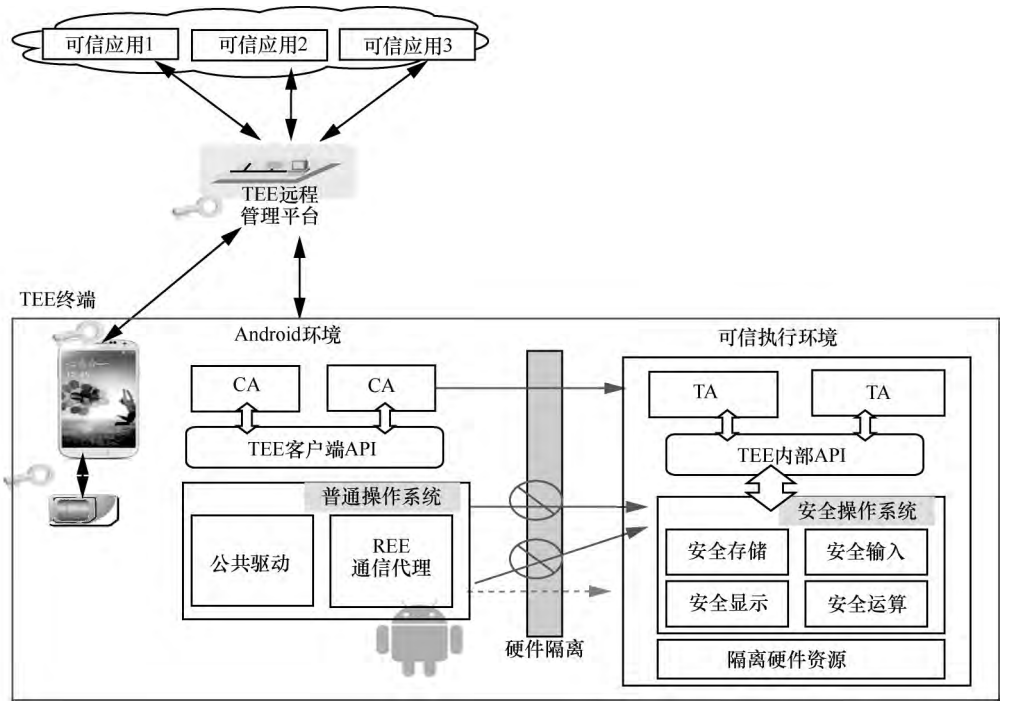


图 2　基于 TEE 的移动终端数据安全框架

分：TEE 远程管理平台（TEEM）和 TEE 终端。

TEEM 是指 TEE 终端远程管理平台，负责 TEE 远程管理和 TA（trusted application，可信应用）的生命周期远程管理，是 TEE 能力入口点和安全应用的汇聚点。TEEM 通过 REE 中的 CA（client application，客户端应用）和 TEE 客户端 API，对 TEE 下发管理指令，实现对 TEE 的管理与授权。TEEM 也接受其他 TAM 所有方的请求，为其在 TEE 里创建相应的安全域，且自己也是一个 TAM 服务器，可以下载和管理自己安全域及 TA。TEEM 主要功能包括：TEE 上的可信应用安全/删除/变更、TEE 安全域创建/删除、TEE 初始化/配置等。

TEE 终端是指基于 TEE 的移动终端，是软件+硬件的全面移动终端安全解决方案。TEE 终端分为两个独立运行的环境区域：REE 和 TEE。REE 作为普通执行环境，即俗称的 Android 系统，是一个开放的系统平台，可运行丰富的 CA，具有通用处理器和硬件环境，负责处理应用业务逻辑和 UI；运行 TAM 业务，负责与 TEEM 进行通信，同时具有访问 TEE 的能力，作为可信任执行环境，相当于硬件上隔离安全区域，加载安全专用系统，具有专用处理器和硬件环境，负责运行签名的 TA，负责数据安全存储（如根密钥、指纹等密钥存储）、数据安全算法（如加解密）、数据隔离和数据安全访问等。

4　业务方案

基于 TEE 的移动终端数据安全框架实现了可信应用加载、应用安全访问、数据安全存储（如指纹存储）、数据安全校验（如指纹校验、隐私数据加解密保护等）、数据安全算法等，具备安全性、可靠性、可维护性，广泛应用于移动支付、隐私数据保护、权限管控、移动办公等业务场景。

4.1　典型 TEE 移动支付方案

现有移动支付方案中，移动支付应用主要部署在 REE，移动应用密钥主要存储在后台系统中，属于应用级保护，密钥容易伪造、泄露。典型 TEE 移动支付方案中，移动支付应用作为可信应用部署在 TEE 中，并在 TEE 中完成密钥存储、加密服务等功能，密钥保护等级属于芯片级保护。

如图 3 所示，基于 TEE 的移动支付方案中，TEE 主要完成可信应用（移动支付 TA）的安装部署、安全访问，并通过生物识别、加密服务、可信 UI 为移动支付业务提供安全防护。

典型 TEE 移动支付方案的关键流程如下。

（1）可信应用安装部署

步骤 1　通过 TEEM 下发指令给 REE，REE 承载的 CA 通过代理访问 TEE，并创建 SD。

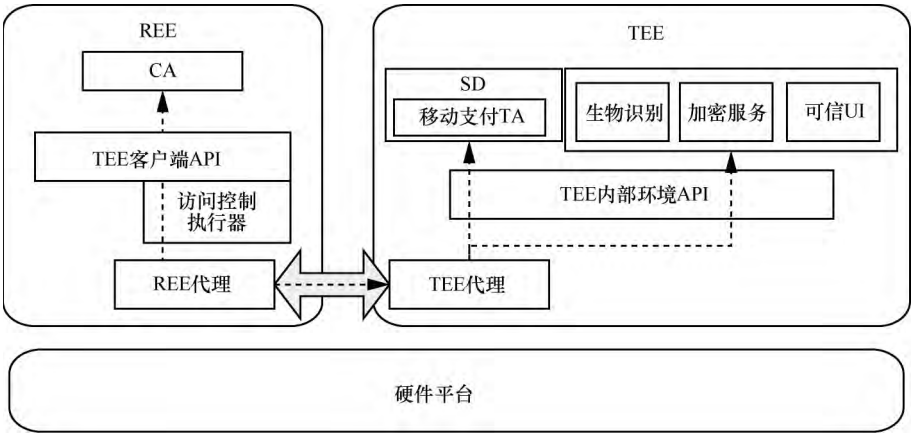


图 3　典型 TEE 移动支付方案

步骤 2　通过向 TEE 的 SD 下载安装移动支付业务的可信应用。

步骤 3　也可在终端出厂时以预置的方式完成安装部署。

(2)可信应用安全访问

步骤 4　REE 承载支付 CA，当支付时调用 TEE 进行身份认证。

(3)TEE 安全防护

步骤 5　通过生物识别，如指纹存储，向支付应用提供身份认证。

步骤 6　通过安全算法，向支付应用、隐私数据提供可信的加密服务。

步骤 7　通过可信 UI，提供可信的安全输入。

4.2　创新型 TEE 指纹动态调整支付额度方案

创新型 TEE 指纹动态调整支付额度方案是在典型 TEE 移动支付方案基础上，进行创新改进。与典型 TEE 移动支付方案一样，在创新型 TEE 指纹动态调整支付额度方案中，移动终端也可在 TEE 部署，运行可信任的移动支付可信应用，并录入指纹作为身份认证的密钥。该方案最大的优点和创新点是：利用移动终端 TEE 可录入多个指纹的功能，为不同指纹关联不同授权额度，每个指纹对应一个额度，所有限制额度的处理都在终端侧 TEE 中完成，与后台无关，从而弥补通过系统侧限制额度导致的改造大、密码记忆难等方面的不足。创新型 TEE 指纹动态调整支付额度方案支持多指纹多额度的动态调整，并可隐秘地启动报警流程等。

(1)指纹授权级别登记流程

指纹授权级别登记流程如图 4 所示。

步骤 1　用户打开移动支付 CA，触发指纹授权级别登记的功能，选择一种级别(正常级别、限制级别 1、限制级别 2 等)后，CA 调用 Android 的指纹 API，提示用户输入对应级别的指纹，用户输入指纹。

步骤 2　Android 对指纹验证成功后，CA 通知 TA 本次指纹对应的授权级别。

步骤 3　TA 调用 TEE 的指纹功能，获取最后一次成功验证的指纹 ID，并把指纹 ID 与对应的授权级别进行绑定登记，绑定关系存储在 TEE 中。用户可继续执行步骤 1，针对不同的授权级别录入不同的指纹。

(2)指纹使用流程

指纹使用流程如图 5 所示。

步骤 1　用户打开移动支付 CA，在未进入应用主功

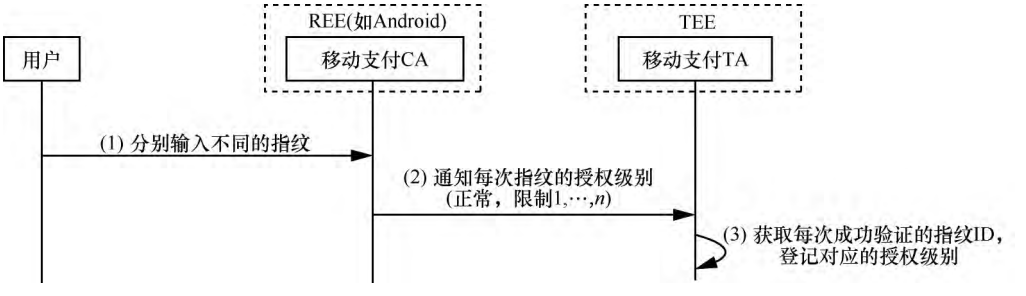


图 4　指纹授权级别登记流程

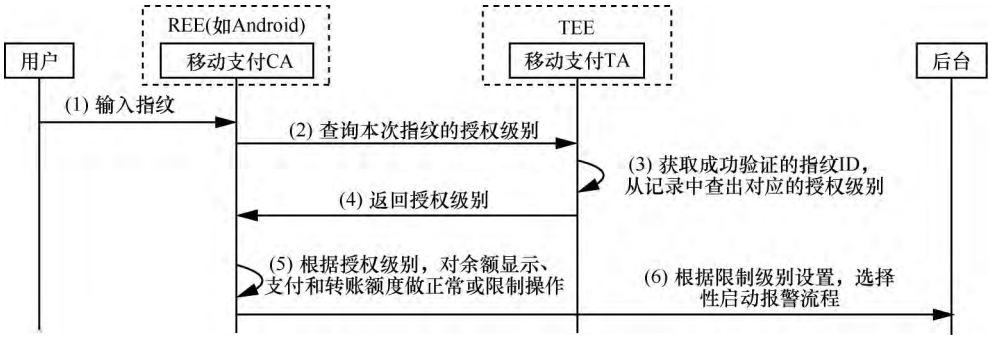


图 5　指纹使用流程

能界面之前，应用调用 Android 的指纹 API，提示用户输入指纹，用户输入指纹。

步骤 2　Android 对指纹验证成功后，应用向 TA 查询本次指纹对应的授权级别。

步骤 3　TA 调用 TEE 的指纹功能，获取最后一次成功验证的指纹 ID，并从之前的登记记录中，根据指纹 ID 查出对应的授权级别。

步骤 4　TA 向应用返回授权级别。

步骤 5　应用根据授权级别，对余额显示、支付和转账额度进行正常或限制的操作。

步骤 6　应用根据限制级别中的预先设置，选择性地向后台发出报警通知。

5　核心代码实现

基于 TEE 的移动数据终端安全的核心代码实现逻辑关系如图 6 所示，关键实现逻辑包括：TEE 初始化、TEE 共享内存申请及分配、TEE 私有内存数据的保存（如保存指纹数据、保存指纹对应的支付额度、权限管控等隐私数据）、TEE 私有内存数据的读取、TEE 私有内存数据处理（如指纹验证、权限验证等）。

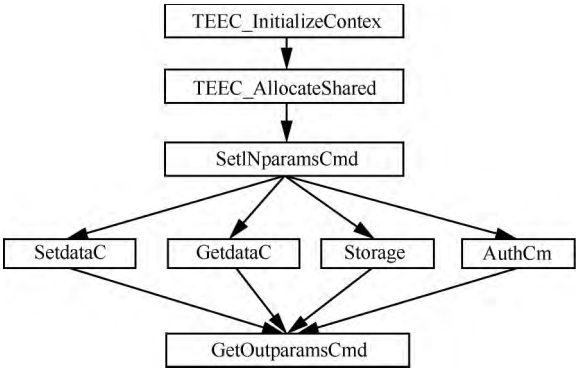


图 6　代码实现逻辑关系

（1）CA 侧函数调用

```
/*CA 初始化 TEE*/
TEEC_InitializeContext(NULL, context);
/*CA 申请 TEE 的共享内存，共享内存是 CA 和 TA 之间传递数据的“池”*/
TEEC_AllocateSharedMemory(context,SharedMem0);
/*CA 设置具体某个命令的输入字段，输入字段放置在共享内存 */
TEEC_InvokeCommand(&session,
FingerprintSetINparamsCmd, &operation, &origin);
/*CA 设置具体某个命令的输出字段，输出字段放置在共享内存 */
TEEC_InvokeCommand(&session,
FingerprintGetOutparamsCmd, &operation, &origin);
/*CA 调用 TA，保存指纹 */
TEEC_InvokeCommand(&session,
FingerprintStorageCmd, &operation, &origin);
/*CA 调用 TA，验证指纹 */
TEEC_InvokeCommand(&session,
FingerprintAuthenticateCmd, &operation, &origin);
/*CA 调用 TA，保存私密数据，如指纹对应的支付额度、权限管控等 */
TEEC_InvokeCommand(&session,
FingerprintSetCAdataCmd, &operation, &origin);
/*CA 调用 TA，获取私密数据 */
TEEC_InvokeCommand(&session,
FingerprintGetCAdataCmd, &operation, &origin);
```

（2）TA 侧函数调用

```
①/* 指纹存储 */
StorageCmd()
```

```
{
    /* 获取 TA 私有内存的全局变量指针 fpdata，fpdata 即 CA 送到 TA 的数据/*
    /*fpdata->fingerprint 表示 CA 送来的指纹，变量 fpdata->appid 表示 CA 送来的应用 ID/*
    FingerprintParams* fpdata = NULL;
    fpdata =(FingerprintParams*)TEE_GetInstanceData();
    /* 根据摘要算法确定 size_hash 和 block，对 CA 送来的指纹进行摘要计算，求出 hash*/
    TEE_AllocateOperation (&operation, HASH_ALOGORITHMID, TEE_MODE_DIGEST, *size_hash);
    …
    TEE_DigestDoFinal(operation, fpdata->fingerprint + i*block, remainder, hash, (size_t*)size_hash);
    /* 用应用 ID 和指纹摘要散列构造摘要的唯一存储文件名，保存摘要 */
    filename=fpdata->appid+hash
    TEE_OpenPersistentObject(TEE_OBJECT_STORAGE_PRIVATE, filename, strlen (filename), TEE_DATA_FLAG_ACCESS_WRITE, (&persistent_storageobj));
    TEE_WriteObjectData (persistent_storageobj, hash, size_hash);
}
②/* 指纹验证 */
AuthCmd()
{
    /* 获取 TA 私有内存的全局变量指针 fpdata，fpdata 即 CA 送到 TA 的数据/*
    /*fpdata->fingerprint 表示 CA 送来的指纹，变量 fpdata->appid 表示 CA 送来的应用 ID/*
    FingerprintParams* fpdata = NULL;
    fpdata=(FingerprintParams*)TEE_ GetInstanceData();
    /* 根据摘要算法确定 size_hash 和 block，对 CA 送来的指纹进行摘要计算,求出 hash1*/
    TEE_AllocateOperation (&operation, HASH_ALOGORITHMID, TEE_MODE_DIGEST, *size_hash1);
    …
    TEE_DigestDoFinal(operation, fpdata->fingerprint + i*block, remainder, hash1, (size_t*)size_hash1);
    /* 用应用 ID 和指纹摘要 hash1 构造摘要的唯一存储文件名，取出存储的指纹摘要 */
    filename=fpdata->appid+hash1
    TEE_OpenPersistentObject(TEE_OBJECT_STORAGE_PRIVATE, filename, strlen(filename), TEE_DATA_FLAG_ACCESS_READ, (&persistent_storageobj));
    TEE_ReadObjectData(persistent_storageobj, hash2, *size_ hash, &count);
    /* 对比 hash1 和 hash2，如果相等则通过认证 */
}
③/* 保存私密数据，如指纹对应的支付额度、指纹对应的权限、隐私数据等 */
SetdataCmd()
{
    /* 获取 TA 私有内存的全局变量指针 fpdata，fpdata 即 CA 送到 TA 的数据 */
    FingerprintParams* fpdata = NULL;
    fpdata =(FingerprintParams*)TEE_GetInstanceData();
    /* 判断 fpdata 中需要保存的字段，判断依据：要保存的字段长度不为 0，不保存的字段长度为 0*/
    if(fpdata->字段 1 长度 ! = 0){
    TEE_OpenPersistentObject (TEE_OBJECT_STORAGE_PRIVATE, 字段 1, fpdata->字段 1 长度, TEE_DATA_FLAG_ACCESS_WRITE, (&persistent_storageobj));
    TEE_WriteObjectData ( persistent_storageobj , fpdata-> 字段 1, fpdata->字段 1 长度);
}
…
}
④/* 获取私密数据 */
GetdataCmd()
{
    /* 获取 TA 私有内存的全局变量指针 fpdata，fpdata 即 TA 送回 CA 的数据 */
    FingerprintParams* fpdata = NULL;
    fpdata =(FingerprintParams*)TEE_ GetInstanceData();
    /* 判断 fpdata 中需要保存的字段，判断依据：要保存的字段长度不为 0，不保存的字段长度为 0*/
    if(fpdata->字段 1 长度 ! = 0)
    {
    TEE_OpenPersistentObject(TEE_OBJECT_
```

```
STORAGE_PRIVATE, 字段 1, fpdata->字段 1 长度, TEE_
DATA_FLAG_ACCESS_READ, (&persistent_storageobj));
        TEE_ReadObjectData(persistent_storageobj, fpdata->
字段 1, fpdata->字段 1 长度,
        &count);
        }
        …
    }
    ⑤/*CA 送数据进 TA */
    SetINparamsCmd(uint32_t      nParamTypes,TEE_Param
pParams[4])
    {
        /* 获取 TA 私有内存的全局变量指针 fpdata,
fpdata 保存 CA 送进 TA 的数据 */
          FingerprintParams* fpdata = NULL;
          fpdata=(FingerprintParams*)TEE_GetInstanceData();
        /* 把共享内存中的字段送进 TA 的变量 fpdata
中,pParams 即共享内存 */
        switch(pParams[0].value.a)
        {
         case 字段 1:
           {
              fpdata->字段 1 长度 = pParams[2].value.a;
              fpdata->字段 1: = TEE_Malloc(fpdata->字
段 1 长度, 0);
              TEE_MemMove(fpdata->字段 1, pParams[1].
memref.buffer, fpdata->字段 1 长度);
              break;
         }
              …
      }
      //将 fpdata 设置为全局变量,供 TA 其他函数调用
      TEE_SetInstanceData((void*)fpdata);
    }
    ⑥*/*TA 送数据回 CA */
    GetOutparamsCmd( uint32_t    nParamTypes,TEE_Param
pParams[4])
    {
        /* 获取 TA 私有内存的全局变量指针 fpdata,
fpdata 保存 TA 送回 CA 的数据 */
        FingerprintParams* fpdata = NULL;
        fpdata =(FingerprintParams*)TEE_GetInstanceData();
       /* 把 TA 的变量 fpdata 的字段送回共享内存中,
pParams 即共享内存 */
        switch(pParams[0].value.a)
        {
        case 字段 1:
          {
              TEE_MemMove (pParams[1].memref.buffer,
fpdata->字段 1, fpdata->字段 1 长度);
pParams[2].value.a = fpdata->字段 1 长度;
              break;
          }
          …
        }
    }
    ⑦/* 安全算法 */
    enum TEE_CRYPTO_ALGORITHM_ID
    {
    TEE_ALG_AES_ECB_NOPAD =  0x10000010, /**<算
法:AES_ECB_NOPAD */
    TEE_ALG_DES_CBC_NOPAD = 0x10000111, /**<算
法:DES_CBC_NOPAD */
    TEE_ALG_RSA_NOPAD = 0x60000030, /**<算法:
RSA_NOPAD */
    TEE_ALG_MD5 = 0x50000001, /**<算法:MD5 */
    TEE_ALG_SHA1 = 0x50000002, /**<算法:SHA1 */
    TEE_ALG_SHA256 = 0x50000004, /**<算法:SHA256 */
      …
    };
```

6 结束语

在指纹+TEE 方案中,将指纹存储(密钥存储)、指纹验证(密钥验证)和可信支付应用加载到基带芯片的 TEE 中处理,有助于提升数据安全处理环境的安全级别至 CC EAL2+,是当前移动终端数据安全的主流选择。未来伴随着移动支付技术的迅猛发展和安全需求的不断升级,运营商着重考虑结合 SIM 卡的 SE 安全能力,通过指纹+TEE+SE 方案,将数据安全处理环境的安全级别从 CC EAL2+提升至 CC EAL5+。指纹+TEE+SE 方案还有助于运营商管控 SIM 卡,并提供 TEE+SE 的差异化安全能力,形成机卡一

体化的移动终端数据安全解决方案，这必将是未来运营商在移动终端数据安全上的重要布局。

参考文献：

[1] 张大伟, 郭烜, 韩臻. 安全可信智能移动终端研究[J]. 中兴通信技术, 2015, 21(5): 39-44.
ZHANG D W, GUO X, HAN Z. Security and trusted intelligent mobile terminal[J]. ZTE Technology Journal, 2015, 21(5): 39-44.

[2] 郭茂文. 基于 FIDO 协议的指纹认证方案研究 [J]. 广东通信技术, 2016, 36(4): 2-5.
GUO M W. Research on fingerprint authentication scheme based on FIDO protocol[J]. Guangdong Communication Technology, 2016, 36(4): 2-5.

[3] 国炜, 王宗岳. 移动终端安全隔离技术分析 [J]. 移动通信, 2016, 40(21).
GUO W, WANG Z Y. Analysis on security isolation techniques on mobile terminal[J].Mobile Communications, 2016, 40(21).

[4] 范冠男, 董攀. 基于 TrustZone 的可信执行环境构建技术研究[J]. 信息网络安全, 2016(3).
FAN G N, DONG P. Research on trusted execution environment building technology based on TrustZone [J]. Netinfo Security, 2016(3).

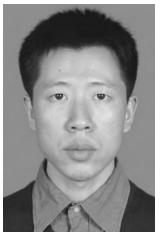
[5] 纪祥敏, 赵波, 陈璐, 等. 基于信任扩展的可信云执行环境[J]. 华中科技大学学报(自然科学版), 2016, 44(3): 105-109.
JI X M, ZHAO B, CHEN L, et al. Trusted cloud execution environment based on trust extension [J]. Journal of Huazhong University of Science and Technology(Nature Science Edition) , 2016, 44(3): 105-109.

[6] 包依勤. TrustZone 技术在 Android 系统中的安全性研究[J]. 物联网技术, 2015(10): 70-73.
BAO Y Q. Research on the security of TrustZone technology in Android system [J]. Internet of Things Technologies, 2015(10): 70-73.

[作者简介]

陈淑珍(1985–)，女，中国电信股份有限公司广州研究院工程师，主要研究方向为物联网、安全、测试技术等。

杨敏维(1972–)，男，中国电信股份有限公司广州研究院高级工程师，主要研究方向为物联网、安全、测试技术等。



何峣(1977–)，男，中国电信股份有限公司广州研究院高级工程师，主要研究方向为物联网终端通信、安全、测试技术等。