**OPTEE安全存储文件解密软件**

设

计

说

明

书

# 1、前言

## 1.1 文档说明

编写本文档的目的：介绍TrustZone技术和OP-TEE可信执行环境的设计特点，并详细分析OP-TEE的安全存储框架模型，最后根据以上信息设计OPTEE安全存储文件恢复软件。

## 1.2 项目目标

项目目标：通过开发安全存储恢复软件，验证我们对OPTEE安全存储原理探究的正确性，后续将在此基础上，研究可信计算中保护数据安全的关键技术点以及相关优化方向。

## 1.3 方案设计依据

GB9566—88《计算机软件开发规范》

GB8567—88《计算机软件产品开发文件编制指南》

# 2、软件设计背景

## 2.1 研究背景

随着互联网时代蓬勃发展和5G技术兴起，嵌入式设备得到越来越广泛应用。与此同时，嵌入式设备所处应用环境变得越来越复杂，各种信息安全攻击技术层出不穷，用户面临着十分严重的安全威胁。因此，安全性设计已经成为嵌入式开发中需要重视的技术要点。

系统数据安全可分为两部分：数据存储安全和数据传输安全，而嵌入式设备所处环境往往不可控，因此如何实现设备数据安全存储保证，成为嵌入式系统安全设计难题。目前一个主流设计方案是：在嵌入式设备上构建一个可信执行环境（Trusted Execution Environment，TEE），并在此基础上，通过软硬件结合方式为用户提供安全可靠的数据存储保障。

在这种方案下，如何构建一个有效的可信执行环境，成为大家首要研究目标。在一个可信执行环境中，核心在可信计算基（Trusted Computing Base，TCB），它由可信硬件基和可信软件基两部分共同构成，二者协同工作为系统提供一定程度的安全保障，系统安全可信度构建在可信计算基安全可信度之上。

可信硬件基包括安全处理器及其相关硬件安全技术，通过隔离、加密、状态刷新等方式，提供基于硬件保障的认证性、机密性、完整性保护等安全特性。硬件保护机制具有很好的稳定性，很难被改变，不像软件实现的安全保护机制容易被篡改。硬件保护机制也不像软件保护机制那样容易被绕过，因为它从嵌入式系统的最底层组件开始进行保护。最后，系统使用专用硬件实现安全保护功能，可以显著降低可信计算性能开销，使得可信执行环境应用具有可行性。

硬件安全技术在工业界得到了广泛应用，其中代表性应用有：IBM 4765协处理器通过物理隔离、软件认证和防篡改攻击进行保护，ARM处理器的TrustZone技术将系统资源划分为可信组件和不可信组件两类实现了硬件隔离，英特尔处理器的SGX（Software Guard extension）技术则注重于减小可信软件基和严格权限访问控制，来保护可信执行区域（Enclaves）中代码和数据安全。TrustZone和SGX技术主要区别在于：TrustZone中可信应用信任整个可信操作系统，而SGX中可信应用只信任自己和Intel的处理器（将操作系统排除在外）。

在嵌入式领域，ARM处理器因为低功耗和高实时性等特点，从而占据了绝对市场垄断地位，而基于TrustZone技术的TEE可信执行系统，则成为构建相应嵌入式TEE环境所必需的可信软件基。TEE可信执行系统为可信应用（Trusted Application，TA）提供一系列安全服务接口，从而使TA能在一个安全隔离环境中使用敏感密钥和数据。在工业界，各大移动设备巨头开发自有的TEE可信执行系统，如高通开发的QSEE、华为开发的鸿蒙、苹果开发的Secure Enclave等，出于安全因素考量，故这些系统都是闭源系统。在学术界，许多学者在Linaro组织开发的OP-TEE[5]、赫尔辛基大学开发的OPEN-TEE[6]以及谷歌公司开发的Trusty[7]等开源系统的基础上展开研究，OP-TEE因为具有完整的开发工具链，项目更新速度快，文档较为齐全而广受关注。

表2.1 基于TrustZone的TEE可信执行系统以及系统设计者

|  |  |
| --- | --- |
| 系统设计者 | TEE系统名称 |
| 高通 | QSEE |
| 华为 | 鸿蒙 |
| 苹果 | Secure Enclave |
| Linaro组织 | OP-TEE |
| 赫尔辛基大学 | OPEN-TEE |
| 谷歌 | Trusty |

另一个主要研究目标是安全存储系统设计、实现和形式化证明，主要通过云存储、虚拟化、安全元件、可信执行环境等技术来提高数据存储的安全级别，从而保证系统中的数据具有机密性、完整性、可用性、不可否认性等一系列安全属性。嵌入式安全存储系统要在有限计算资源上，实现运行开销和功耗的控制，又因嵌入式设备普遍需要进行广泛部署，从而面临着巨大的物理攻击威胁和挑战，因此系统设计需要考虑很多方法的问题，如：密钥的生成与保护、加密算法的选择与优化、存储硬件的设计与应用等。需要注意的是，TrustZone技术侧重于通过建立可信执行环境来保证运行时数据的安全，而数据静态安全还需要结合安全元件(Security Element，SE)等硬件进行保护。

下一节，我们将详细分析OPTEE的安全存储模型，为第三章描述软件开发设计思想做好准备。

## 2.2 OPTEE安全存储框架

OPTEE为用户提供数据安全存储服务，用户可使用此功能来保存敏感数据、密钥等关键信息。OPTEE安全存储实现遵循Global Platform规范，以保证存储数据的机密性和完整性，以及修改存储的操作的原子性。用户只需要调用Global Platform规范的安全存储接口，就可以实现对私有数据的保护。安全存储中的数据加解密过程均在OPTEE中完成，且相关密钥的生成和保护也在OPTEE中，这样就能确保数据的安全性。

OPTEE的安全存储有两种存储实现：基于普通世界的文件系统（Rich Execution Environment， REE）存储，基于eMMC(Embedded Multi Media Card）的重放保护分区（Replay Protected Memory Block，RPMB）存储。针对用户不同安全存储需求，开发者可以在sub/config.mk配置文件中，进行存储方式配置。

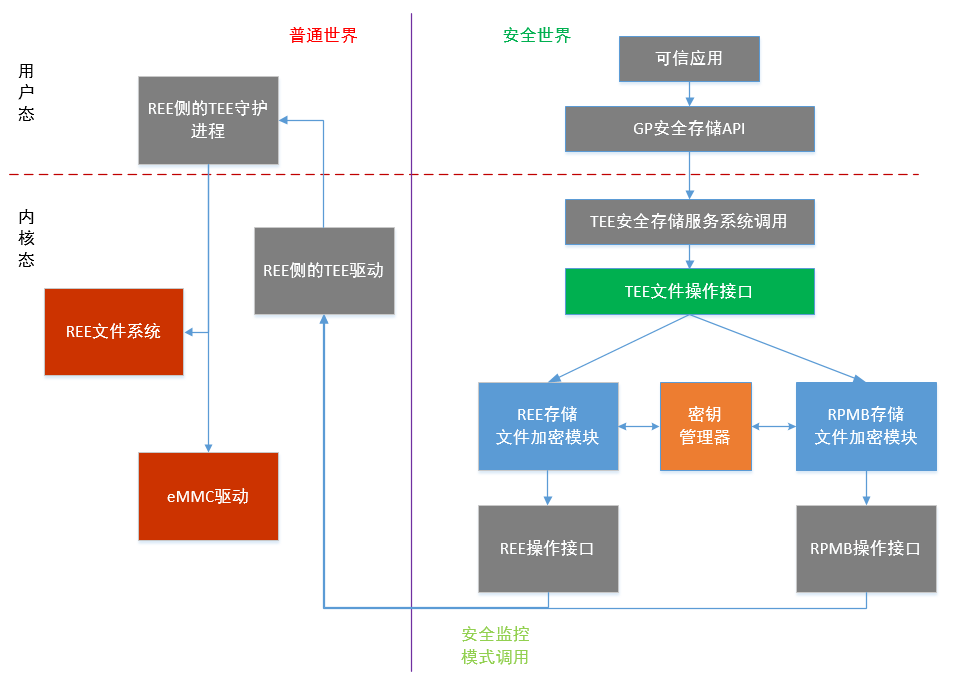


图2.1 OPTEE安全存储框架图

OPTEE安全存储框架如图2.1所示，其中主要功能模块有：

1. 安全世界中的加密模块：主要负责安全存储对象加密和完整性检验操作；
2. 普通世界中的存储模块：主要负责加密后安全存储对象物理读写过程；
3. 安全世界中的密钥管理器：为OPTEE系统提供应用于各个层次、不同功能的密钥。

在OPTEE中，可信应用程序TA调用安全存储服务详细过程如下：

1. 可信应用程序TA调用Global Platform规范的安全存储接口获取服务；
2. GP安全存储接口将TA安全服务请求转换为相应的TEE安全存储服务系统调用；
3. 系统调用获取TA的会话ID以及运行上下文进行权限检查，权限检查无误后使用TEE文件操作接口实现安全数据读写操作；
4. TEE文件操作接口依据系统底层不同的存储方式，调用TEE内核态中不同的加密模块。加密模块主要负责安全数据加解密操作，通过密钥管理器获取操作密钥，并将操作结果保存在安全存储对象中；
5. 由于OP-TEE无法直接操作普通世界中REE文件系统或eMMC的RPMB分区，因此加密模块为了实现安全存储对象的读写操作，需要通知REE侧的存储模块执行安全存储对象持久化操作。加密模块使用了安全监控模式调用（Security Monitoring Call），通过TEE驱动转发RPC（Remote Procedure Call）操作，从而通知REE侧的TEE守护进程执行相应请求；
6. TEE守护进程调用存储模块程序，实现一系列安全存储对象持久化操作，例如：安全存储文件的创建、打开、读取、写入、重命名、删除等。

# 3、软件详细设计

本软件工具设计原理是逆向工程思想，分析OPTEE安全存储模型以及REE存储执行过程，据此构建相应的数据解密工具。

通过2.2节对OPTEE安全存储框架分析，我们可以得知存储框架主要由三个模块组成：密钥管理器模块、存储模块、加密模块。因此我们设计的数据解密工具，也相应逆向设计了对应三个模块: 密钥管理器模块、存储文件解析模块、文件数据解密模块，分列三小节进行介绍。最后一小节我们介绍本软件的运行方法和测试结果。

## 密钥管理器模块

密钥管理器作为安全存储系统中的一个重要组成部分，为系统提供应用于各个层面的加密密钥，从而实现系统身份验证、数据机密性和完整性保护等功能。在OPTEE中，密钥管理器主要通过构建密钥链的方法获取密钥。密钥链基本设计思想：以硬件根密钥HUK（即硬件唯一密钥，Hardware Unique Key）作为信任根基点，链式派生与保护系统中所有密钥。确保密钥派生算法单向性是保证密钥链安全性关键所在，单向性保证了派生密钥被访问时，根密钥安全性能得到足够的保护。在此基础上，密钥链分别采用硬件、操作系统、安全应用等各个体系层次中信息常量来获取多级密钥，实现了不同功能、不同安全应用之间的密钥隔离，大大提高了攻击者攻击难度。

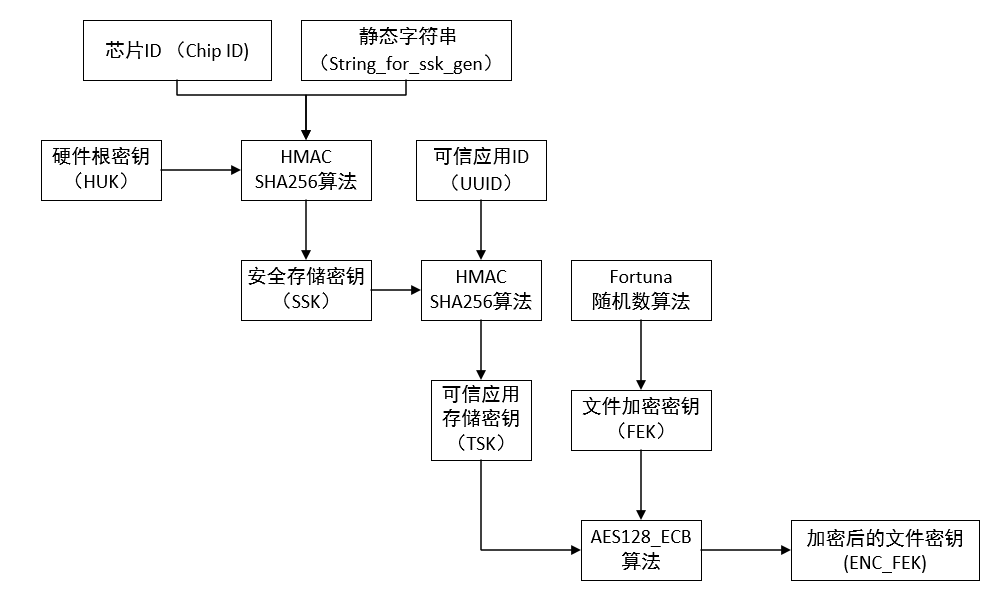


图3.1存储系统密钥链构建过程示意图

本软件的密钥管理器，实现了OPTEE安全存储系统密钥链的相关密钥构建。密钥管理器中各级密钥的含义与作用是：

1. 硬件根密钥（Hardware Unique Key，HUK）： TEE建立可信系统的信任根源，需要安全地存储在硬件中，且最好是一机一密，TEE系统中的其他密钥都应由根密钥进行派生和保护；
2. 安全存储密钥（Secure Storage key，SSK）：在TEE系统中，不同的系统功能有各自不同的密钥，除了安全存储密钥，还有身份认证密钥、完整性度量密钥等；
3. 可信应用存储密钥（TA Storage key,TSK）：不同的可信应用，对应不同的可信应用存储密钥；
4. 文件加密密钥（File Encryption key，FEK）：不同的安全存储文件，对应不同的文件加密密钥；
5. 加密后的文件加密密钥（Encryption FEK，ENC\_FEK）：文件加密密钥需要加密后持久化保存在硬件中。

实现密钥管理器，最终目的是为了获取解密数据文件所需的密钥FEK，构建过程可以概述为以下公式：

1. SSK = HMAC\_SHA256(HUK，CHIP\_ID || STR\_FOR\_SSK\_GEN)
2. TSK = HMAC\_SHA256(SSK,UUID)
3. FEK = AES\_ECB128\_Decrypt (TSK, ENC\_FEK)

使用密钥管理器构建密钥链上的所有密钥，在这个过程中需要使用各种属性，表3.1是我们从OPTEE项目中提取到的构建密钥链所需各项属性值。

表3.1 构建密钥链所需属性值表

|  |  |  |
| --- | --- | --- |
| 属性名 | 属性所在文件路径 | 属性值 |
| 硬件根密钥：HUK | core/arch/arm/kernel/otp\_stubs.c | 16个字节的0 |
| 芯片ID： chip\_id | core/arch/arm/kernel/otp\_stubs.c | 8个“BEEF”顺序组成的  32字节字符串 |
| 生成SSK的字符串：str\_for\_ssk\_gen | core/tee/tee\_fs\_key\_manager.c | “ONLY\_FOR\_tee\_fs\_ssk” |
| 索引文件的可信应用ID: dirfdb\_uuid | core/tee/tee\_fs\_key\_manager.c | 1个字节的0 |

## 存储文件解析模块

当OPTEE选择REE存储方式时，会将安全存储对象数据加密存储到REE文件系统下的安全存储文件中，默认情况下这些文件会被保存在/data/tee目录下。

安全存储文件类型按功能可分为两类：

1. 数据文件：一个从0号开始依次递增的数字编号文件，除了0号数据文件保存可信TA的索引信息，其余数据文件均保存了一个安全存储对象数据。
2. 索引文件：一个名为“dirf.db”的文件，它保存了所有数据文件的目录信息结构体dirfile\_entry。可以把索引文件视作一个对象列表数据库，其中保存了存储对象ID、创建对象的可信应用ID、数据文件编号等信息。

因此，解密数据软件的存储文件解析模块，需要依次解析索引文件和数据文件，从中获取各个部分加密数据，将其提供给解密模块解密，最后获取解密后的安全存储对象进行存储。解析模块基本流程如下：

1. 调用解密模块解密索引文件；
2. 解析索引文件，从中找到安全存储对象的索引数据；
3. 根据索引数据，调用解密模块解密数据文件；
4. 解析数据文件，从中找到安全存储对象的实际数据。

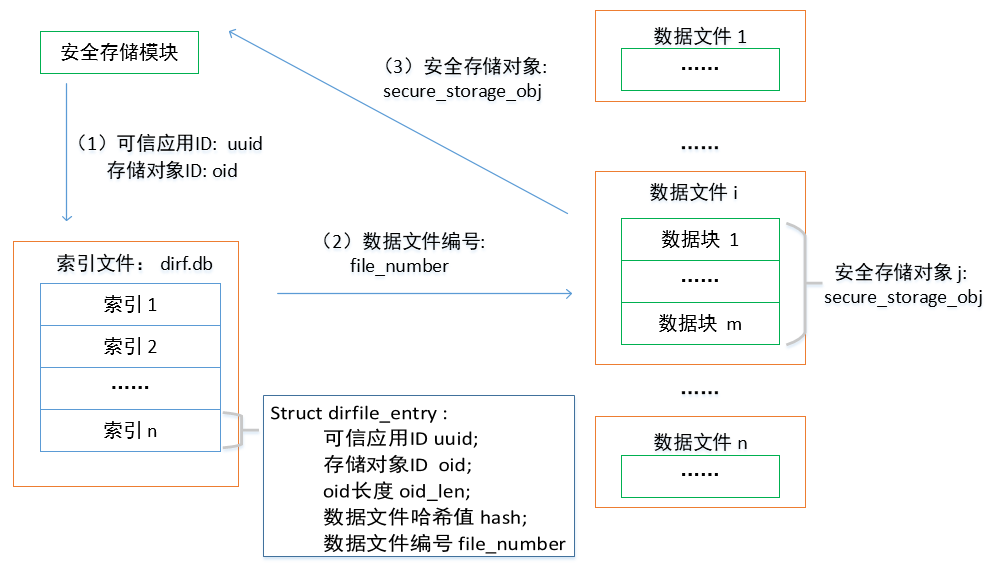


图3.2 存储解析模块基本流程

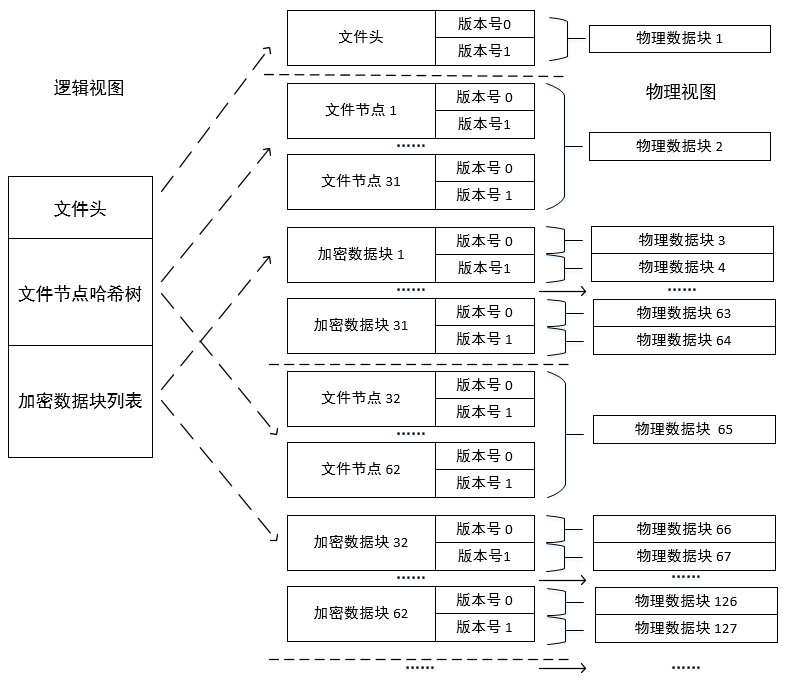


图3.3 物理数据块为4KB时，安全存储文件的存储格式

索引文件和数据文件采用了相同数据存储方式，来保存文件元数据以及索引数据、对象数据，如图3.3中所示，安全存储文件格式由三类结构元素构成：

1. 文件头（tee\_fs\_htree\_image结构体）:保存了整个文件元数据，包括存储数据长度和文件加密相关元素信息；
2. 文件节点（tee\_fs\_htree\_node\_image结构体）:保存加密数据块相关元素，以及文件节点哈希值；
3. 加密数据块:在索引文件中保存了索引列表（索引数据），在数据文件中保存了存储对象数据块列表（对象数据）。

三类存储结构元素均有两份物理持久化数据：版本号0元素和版本号1元素。OPTEE为了实现安全数据的原子操作需求，使用了两个版本数据进行操作：一份活跃版本数据和一份不活跃版本数据。所有更新都是在非活跃版本中完成的，修改期间活跃版本数据不受影响，直到文件根哈希值写入索引文件为止更新完成，一旦原子更新失败，可以轻松实现回滚操作。因此存储解析模块还需要计算存储元素的活跃版本，再去解析相应数据。

## 文件数据解密模块

OPTEE系统为了实现存储对象数据保护，不仅要通过加密方式提供数据机密性保护，也要提供数据完整性验证。OPTEE借鉴了Yan等人提出的存储加密方案,采用了一种数据认证加密模式AES-GCM，不仅可以将加密和认证操作合二为一，也能更好地利用ARM处理器AES加速指令，进一步降低认证加密过程开销。

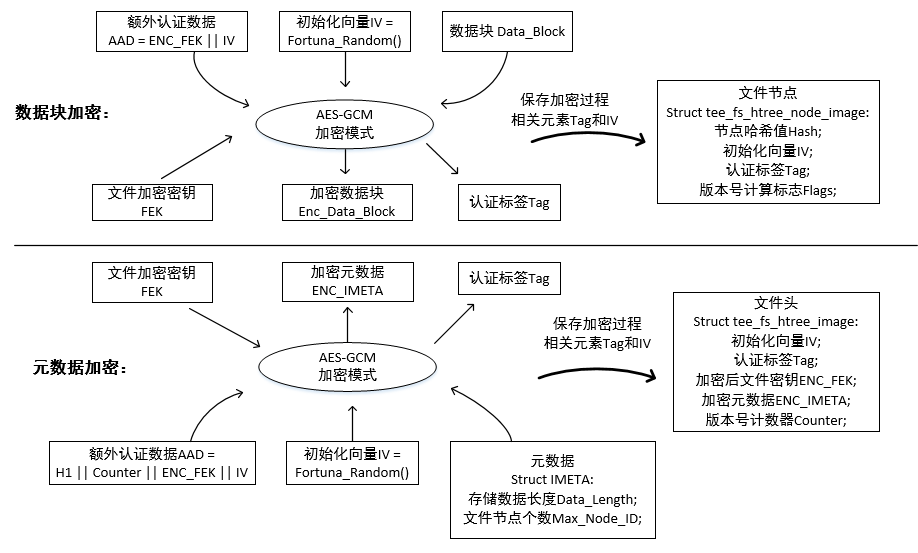


图3.4 REE存储方式中AES-GCM模式加密示意图

AES-GCM模式本质上是在国际先进加密标准AES算法的计数器模式（Counter，CTR）基础上，结合伽罗瓦消息认证码模式（Galois Message Authentication Code，GMAC）实现运算的一种组合模式。在该模式之下，CTR模式负责运算加密功能实现，GMAC模式负责运算认证功能实现。因此给定一个密钥，该模式将明文（Plaintext）加密为密文（Ciphertext），同时使用额外认证数据（Additional Authenticated Data,AAD）生成认证标签（Tag），使用方法形如：(Ciphertext,Tag) = AES\_GCM\_Encrypt(Key, IV,Plaintext,AAD)。

文件数据加密后需要保存相关加密元素Tag和IV，数据块加密时相关元素保存在对应文件节点中，文件元数据加密时相关元素保存在文件头中。文件头tee\_fs\_htre \_image主要负责保存加密后的文件密钥和文件元数据，文件密钥FEK用于解密文件元数据和数据块，文件元数据IMETA保存了文件节点个数Max\_Node\_ID和加密数据块数据长度。

因此文件数据解密模块需要从存储文件解析模块获取相关数据，然后使用AES\_GCM\_Decrypt实现索引文件、数据文件解密。单个文件解密过程如算法3-1所示：

【算法 3-1】单个文件解密过程crack\_secure\_file()

输入：安全存储文件路径：path，可信应用密钥：TSK，哈希树根哈希值：hash

输出：解密后的安全存储文件数据blocks，数据长度length

1. 获取文件头：htree\_image = get\_htree\_image(path,hash)；
2. 获取文件加密密钥：FEK = AES\_ECB\_Decrypt(TSK,htree\_image.ENC\_FEK)；
3. 获取元数据：imeta = AES\_GCM\_Decrypt (FEK, htree\_image.IV, htree\_image.Tag ,htree\_image. ENC\_FEK)；
4. 获取哈希树列表：htree\_nodes = get\_htree\_nodes(path,htree\_image,imeta.length);
5. 获取加密数据块列表：enc\_blocks = get\_blocks(path,htree,imeta.length);
6. 遍历加密数据块列表：for i from 0 to（enc\_blocks.size() - 1） step 1:
   1. 解密得数据块：blocks[i] = AES\_GCM\_Decrypt(FEK,htree[i].IV, htree[i].Tag ,enc\_blocks[i]);
7. 获取数据长度：length = imeta.length；

## 运行方法和测试结果

ReeDataRecovery工具的使用方法是：

./ree\_data\_recovery [storage\_path] [recovery\_path]

其中storage\_path是REE存储方式的安全存储文件存放目录, recovery\_path是用户保存恢复数据对象的路径。

为了完成ReeDataRecovery工具正确性测试，首先需要根据OPTEE 可信应用开发规范和GP TEE接口规范创建使用安全存储功能的可信应用。我们在主机的ubuntu系统上进行可信应用开发，开发时在QEMU上进行调试，最后移植到树莓派3B开发板上。

我们创建了一个可信应用叫MySecureStorage，使用方法是：

my\_secure\_storage [obj\_id1 obj\_id2 ... obj\_idn]

输入是一系列安全存储对象的ID，此时应用会按顺序创建相应的安全存储对象，每一个对象存储的是500个字节0xa1的数据。

然后我们将应用放入OPTEE OS项目中对应位置，修改相应的工程编译文件，之后将整个系统按步骤烧录到Micro SD上。下一步将SD卡装到树莓派3B开发板上，启动树莓派并使用CP2102 USB转串口模块建立主机和树莓派通信，接下来使用minicom命令启动控制台，调用MySecureStorage应用创建安全存储对象。

确认对象创建无误后，将树莓派关机，通过mount命令将SD卡挂载到主机上，将SD卡中/data/tee中所有文件拷贝到主机的任意目录下。最后在主机上通过ReeDataRecovery工具恢复该目录下的数据。

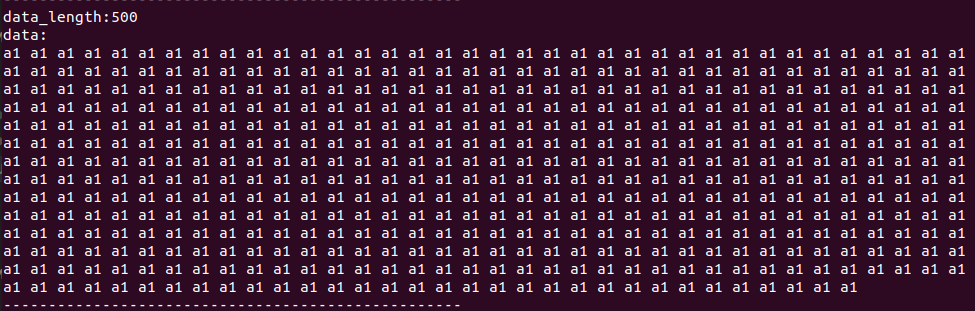


图3.5 实验结果：解密后的数据列表（按十六进制进行输出）

实验结果如上图3.5所示，通过上述实验结果可以确认本软件工具的实现是正确的。