Computer Science and Application 计算机科学与应用, 2016, 6(10), 590-596
Published Online October 2016 in Hans. http://www.hanspub.org/journal/csa
http://dx.doi.org/10.12677/csa.2016.610073

Hans汉斯

Secure Access to Data Transmission for Inter-Component Communication

Lifang Yu, Tianchang Yang, Shaozhang Niu

Beijing Key Lab of Intelligent Telecommunication Software and Multimedia, Beijing University of Posts and Telecommunications, Beijing
Email: 1508552569@qq.com

Received: Oct. 4th, 2016; accepted: Oct. 19th, 2016; published: Oct. 26th, 2016

Abstract

With the Android platform of mobile phone subscribers increasing, the security problem is becoming more serious and receives much concern. The security mechanisms, such as sandbox mechanism, signature mechanism and permission mechanism, are adopted in the Android platform in various ways such as to ensure the security of application, but there are still some serious security issues, such as elevation of privilege attacks. The proposed scheme is to encrypt the transmission data mainly based on the permissions. If there is an elevation of privilege attacks, but the visitors do not have access to the case, then the sensitive data privacy cannot be accessed.

Keywords

Security, Privilege Escalation Attacks, Privacy of Sensitive Data, Encryption

组件间数据传输安全访问设计

余丽芳，杨天长，牛少彰

北京邮电大学智能通信软件与多媒体北京市重点实验室，北京
Email: 1508552569@qq.com

收稿日期：2016年10月4日；录用日期：2016年10月19日；发布日期：2016年10月26日

文章引用: 余丽芳, 杨天长, 牛少彰. 组件间数据传输安全访问设计[J]. 计算机科学与应用, 2016, 6(10): 590-596.
http://dx.doi.org/10.12677/csa.2016.610073

摘 要

基于Android平台的手机用户量逐年增长，随即而来的安全问题也备受关注。Android安全机制中采用了沙箱机制，签名机制，权限机制等各种方式保证应用程序的安全性，但是也存在一些严重安全问题，比如特权提升攻击。本文提出的方案主要是基于权限的基础上，对传输的数据进行加密处理，如果存在特权提升攻击，但是访问者没有权限访问的情况下，则无法访问到隐私敏感数据。

关键词

安全，特权提升攻击，隐私敏感数据，加密

1. 引言

Android 平台是 Google 推出的一款基于 Linux 内核的智能手机操作系统，由 Android 平台在一推出就具有开放性和可扩展性的特性，导致很多移动终端厂商和开发者加入此领域，开发出各种用户喜爱的产品，使得 Android 系统逐渐在智能手机和平板电脑领域占据着大部分市场。也正因为如此，手机安全隐患也越来越多，Android 本身的设计具有良好的安全特性，但这并不意味着 Android 平台不存在安全隐患。

随着最近几年移动手机逐渐成为人们行为生活的一部分，用户的敏感信息也逐渐开始渗透到开放的移动平台，利用 Android 平台存在的安全隐患，恶意攻击者以不正当的手段获取用户或者设备的敏感数据，并滥用此数据做非法操作，非法盈利。隐私泄露大致为：首先，应在运用程序在通信的过程中，经常携带的个人敏感信息，吸引恶意开发者在 Android 应用程序嵌入恶意代码或者应用来窃取敏感数据进行恶意操作。第二，应在运用程序的数据传输过程中恶意拦截意图，使得用户的真正意图转换到恶意的程序，从而导致隐私泄漏。混淆代理人攻击是一类一般出现在 Android 应用程序中的漏洞。当这些漏洞被攻击者触发，脆弱的应用程序会被攻击者利用，泄漏敏感信息，危及 Android 设备上的数据完整性。纯粹的依赖开发者来修复这些漏洞往往是不切实际的。有两个原因：1) 确认每一个漏洞并为其发布补丁对开发者来说是非常耗时的；2) 开发者可能没有足够的经验去妥善的解决问题。

在 Android 平台上众多安全实例的攻击和隐私泄露问题已经是学术研究和公众关注的最近的一个热点，最为突出的有混淆代理人攻击。在 2013 年 USENIX 安全会议中，提出存在于智能手机许多的安全威胁是应用程序组件之间的相互作用的结果，而不是组件内的，并制定针对 Android 平台的良好的静态分析技术[1]。为了防止特权提升组件攻击，Cui X 等人[2]提出了构建调用链的方式，通过分析所有存在的组件的调用，构建抽象语法树和构建控制流图，并构建每个组件的调用链，最后生成调用图，这种方法能知道组件的调用关系，并有效的防范，但是需要分析整个应用程序，实用性不是特别强。Backes, M [3]一文中提出基于 IPC 机制的 Binder 通信过程中，由于无论什么时候 APPS 都可以作为客户端和服务提供者，所以 IPC 机制提供消息和接收建接收消息的相互可信度不够，无法确认发送方身份与接收方身份的情况下将，导致了一定程度的攻击，最为突出的就是混淆代理人攻击，导致隐私数据泄露。本论文实现是在 kernel 层建立相互通信的调用链方式，扩展 Binder 内核模块与 Android 消息处理机制，从而建立可信赖的双方通信，这种方式有效防范混淆代理人攻击，但是付出的代价是修改 Binder 内核模块，实现起来复杂。Nauman M 等人[4]提出一种细粒度的、以用户为中心的权限模型，这个模型允许用户对他们安装的软件进行有选择的授予权限。在这些研究中，基于权限控制对用户数据泄露检测的方式，由于需要用户的频繁参与，被认为是一种不太实用的手段。在参考文献[5]中提出了一种针对恶意应用程序在

Android 平台上特权提升攻击的解决方案：XManDroid 的设计与实施，扩展了 Android 的监控机制，通过 XManDroid 维护系统状态，包含安装的应用程序及它们之间的通信链路，默认的 Android 参考监视器授予一个 ICC 调用时，XManDroid 被调用，并验证是否要求的 ICC 调用，所以它对合谋攻击依然无能为力。对于数据隐私防护方面，MockDroid [6]与 AppFence [7]对于敏感信息可以提供虚假的信息给恶意程序，比如可以实现虚假的位置信息，联系人列表为空等迷惑恶意程序，防止实施非法操作。

以上实现在一定程度上对于特权提升都能实现对组件通信的保护，有些施行效率比较低且实施比较复杂，而且但是如果数据一旦泄露，对正在传输的数据没有进行安全保护，其敏感数据仍然会受到安全威胁。因此，本文将在基于权限的基础上，对于传输数据进行安全处理，所以，即使恶意攻击者得到隐私信息，没有正确的解密方式也无法进行恶意操作。而且相对于以上的各大研究，实现过程中比较方便，而且利用现有的安全技术以及加密技术，实现了开发者所需的 API，在不许需要 root 权限下开发者可以根据需要使用这些 API，并且对真实的数据做了加密处理，本文提出的设计如果数据泄露，在很大程度上恶意攻击者是无法看到用户真实的数据的。

2. 背景知识

2.1. 特权提升攻击

按照文献[8] [9]的定义，特权提升攻击指恶意程序利用其他应用程序未被保护的 API，其中此 API 能够获取另外一个应用的敏感信息，这样恶意程序可以间接获取到所需信息进行非法操作。如图 1：访问应用程序 C 的组件 C1 具有权限 P，应用程序 B 的组件 B1 授予了访问权限 P，可以访问应用程序 C 的 C1 组件，应用程序 A 的组件 A1 没有访问组件 C1 的权限，如果进行访问则会拒绝，但是 A1 能访问未受保护的组件 B1，此时恶意组件可以通过 B1 非法获取 C1 的敏感数据，利用这种权限传递的方式达到自己的目的。

2.2. 数据库加密

在信息处理系统中，密码学的主要应用有两类：数据的通信保护和数据的存储保护，密钥管理是一种读取、修改或验证保护数据的保密代码或数字。密钥与算法(一个数学过程)结合在一起以保护数据。信息安全有赖于密钥管理的有效性，即保证密码的产生，存储，传输和使用的安全性，这就要求对密钥进行有效的管理。数据库加密实行二级密钥管理体制，一级密钥(也称主密钥)存储在安全区域，用它对二级密钥信息加密生成二级密钥(也称为工作密钥)，再用工作密钥和加密算法对数据加密返回给主程序。

3. 数据传输安全访问设计方案

数据传输的安全防护是在 Android Framework 层提供一整套与实现方案。

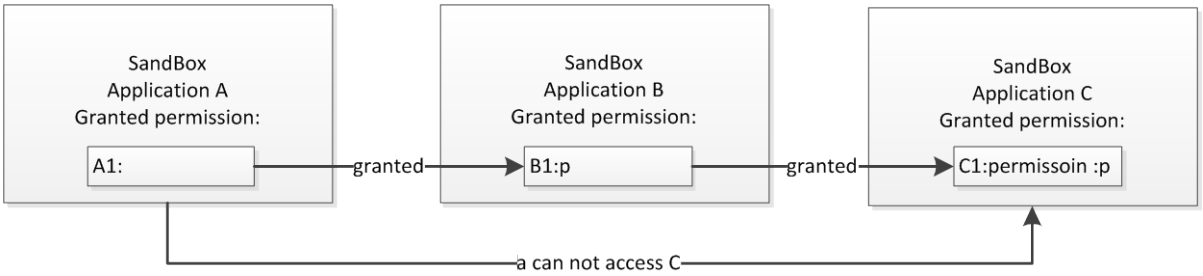

Figure 1. Schematic diagram of privilege elevation attack
图 1. 特权提升攻击示意图

基于已有的数据库加密及 Android 服务的知识，在 Android Framework 层提供一套自身的密钥管理服务 KeyStoreManagerService (自定义的服务类，提供一系列的加密解密等接口类)，并添加到系统 ServiceManager 管理服务类中，用于对数据传输的密钥进行管理及提供数据的加密解密 API。在源码层次 frameworks/base/services/core/java/com/android/server 下添加 KeyStoreManagerService 服务，该服务使用 aidl 创建，实现跨进行通信，具有加密 decrypt，解密 encrypt 及检查权限 checkPermission 三个重要的方法等。

在源码中添加一个系统级别的 ContentProvider，在源码层次 packages/providers 目录下，添加 KeysContentProvider 工程，该工程只有一个用于存储密钥及相关信息的 ContentProvider，内部使用标准的 Sqlite 创建及操作数据库，数据库包含三个字段，分别是 permissionInfo，publicKey，privateKey 三个字段，对应“包名+/+组件名+:+权限”格式的字符串作为唯一标识即 permissionInfo，及两个 blob 类型的二进制公钥数据 publicKey 和私钥数据 privateKey。KeyStoreManagerService 调用自定义类 KeyStoreContentProvider 操作 KeyStoreDataBase 数据库进行密钥的生成，存储以及销毁(如图 2)，并提供 encrypt 数据加密与 decrypt 数据解密 API。

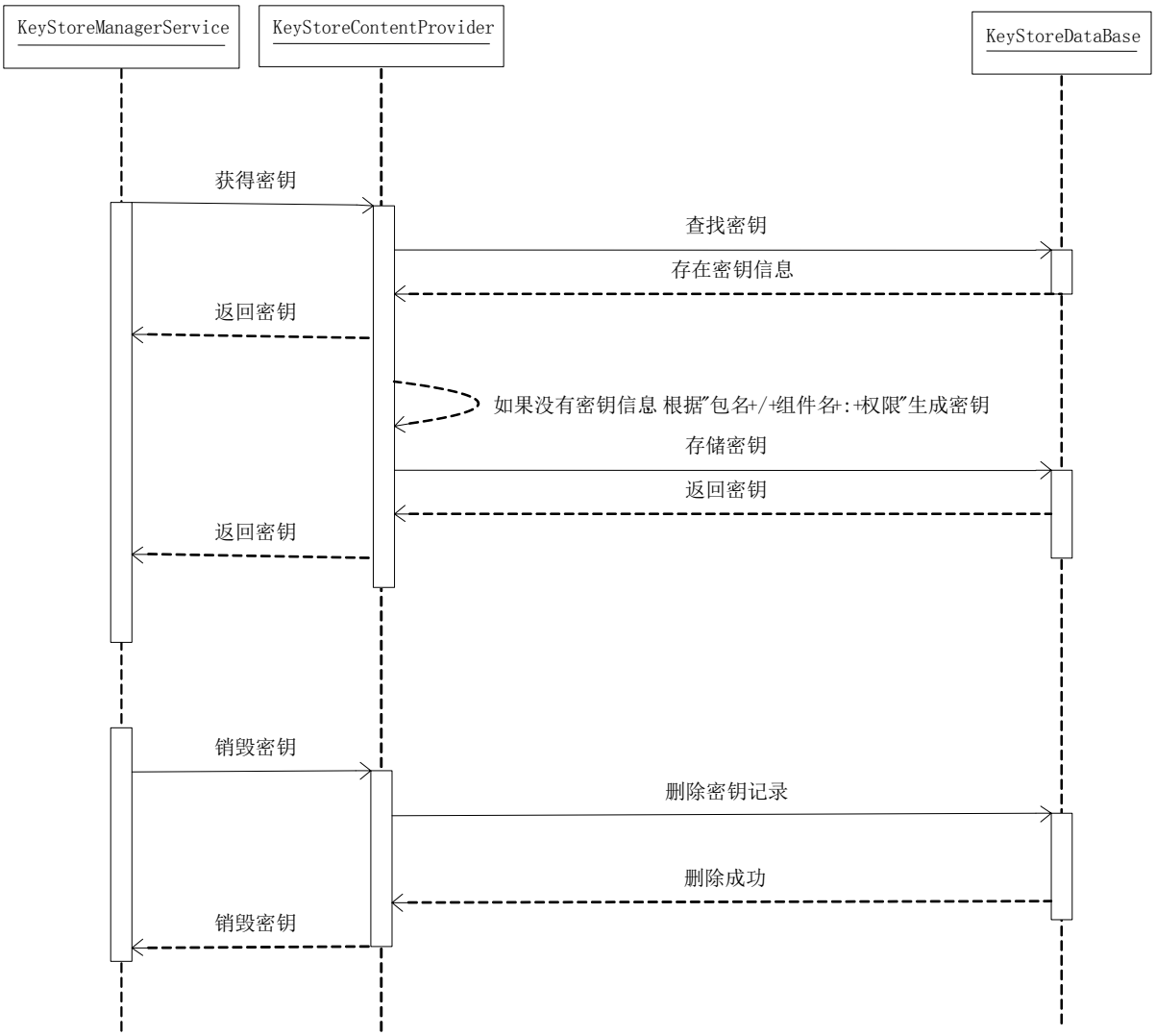

Figure 2. The design of Component Data Secure Communication
图 2. 组件数据安全设计整体图

访问者在调用需要权限访问的组件的时，要求返回隐私或者敏感的数据(如地理位置，个人隐私等)，根据以上介绍的特权提升攻击，不具有访问权限的非法攻击者也可以间接获取到这些敏感数据。为了使用户的敏感数据得到安全防护，本文实施的策略是：组件在具有权限的基础上，如果要求返回数据给访问者，则对返回数据进行加密。Android 系统为了屏蔽进程的概念，抽象的利用不同组件[Acitvity, Service]来表示进程间通信，其中跨进程通信的数据传输是可以通过 Bundle 和 Intent 利用 Binder 机制来实现。被访问者在 put 数据的过程封装在 Bundle 与 intent 中，调用 KeyStoreManagerService 的 encrypt 远程调用，encrypt 调用中首先获取该进程的 uid 和 pid，以及调用方自身组件 token (Activity 在 ActivityManagerService 的唯一标识)，encrypt 调用中根据 token 取得组件在 ActivityManagerService (简称为 AMS，AMS 用于管理所有的 Activity 的生命周期，它同时也管理着系统的 service、broadcast 以及 provider 等)中的信息，判断该调用组件是否具有 permission 属性，如果该组件具有 permission 属性，说明此组件是敏感组件，需要安全防护，所以根据应用的 permission 及组件信息在 KeyStoreManagerService 构建一对公钥私钥，并用“包名+/+组件名+:+权限”作为唯一标识来表示此组件传回的数据是加密数据，需要验证通过后才能解密，同时也作为获取主密钥的唯一标识。数据库采用加密二级密钥管理体制，把主密钥与唯一标识存入数据库。数据调用 encrypt 加密接口，并使用工作密钥对数据进行加密处理并返回。得到的加密数据与唯一标识一起传给调用方，KeyStoreManagerService 获得密钥的过程如图 2 所示。

KeyStoreManagerService 的 decrypt，encrypt 方法中需要传入 token 参数，该参数是组件自身在 ActivityManagerService 的信息的描述，KeyStoreManagerService 的 encrypt 方法是根据 Binder.getCallingUid()获得调用方应用的信息，并根据 token 获得在 ActivityManagerService 中的组件信息，然后根据组件信息在 KeysContentProvider 查找对应的密钥信息，如果没有，就会添加一条密钥信息，并将密钥返回，然后使用该密钥进行数据加密。根据返回的加密数据以及唯一标识传输到访问者，访问者在调用解密 decrypt 方法中会使用 Binder.getCallingUid()获得自身应用的信息，并根据 uid 在 ActivityManagerService 中获得 token 表示访问者的组件信息，在 PackageManagerService 中查询应用是否具有访问对方组件的权限，如果有，就会根据权限及返回信息在 KeysContentProvider 中查找密钥，并根据密钥对数据进行解密，返回解密后的数据和密钥在 KeyStoreContentProvider 中的索引位置。

4. 数据传输安全访问实现

4.1. 基于权限的数据传输加密策略

在 Intent 类中，对 put 类型的方法进行重写，所有 putXXX 类型的方法添加 putXXXSignature 方法，例如 putIntSignature，putStringSignature 等方法，在 putXXXSignature 方法中调用 KeyStoreManagerService 的加密 encrypt 方法，该方法会使用密钥将数据加密后返回，获得加密数据后，填充到 Bundle 中，然后获得调用 encrypt 方法时返回的密钥在数据中的加密索引，格式为：“包名+/+组件名+:+权限”，并使用 put 方法填充进 Bundle 的中，例如:intent.putExtra (“_permission”, “data”)，就完成了数据加密过程的处理。在调用 Intent.putSignature 方法时，需要将自身 token 传入 putSignature 方法中，例如 intent.putSignatureString (activity.getActivityToken(), “data”)，第一个参数是 Activity 在 ActivityManagerService 中的 token，第二个是需要加密的数据。然后该方法中会调用 encrypt 方法，encrypt 方法和 putSignature 方法所传参数必须一致。加密过程中产生的密钥和唯一标识在存储在数据库中，并把加密数据和密钥在数据库中的唯一标识则返回调用者并发送给对方。加密过程如图 3。

1) 在 Intent 填充数据的方法中，调用 KeyStoreManagerService 的加密方法，并将调用该方法的 Activity 的对象的 token 属性传入 encrypt 加密方法中，通过 encrypt 传递到 KeyStoreManagerService 中。

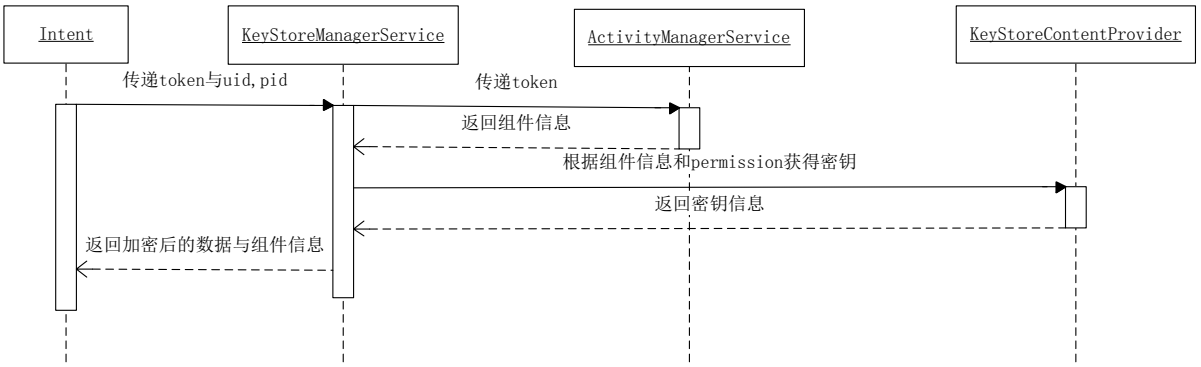

Figure 3. The flow chart of encryption
图 3. 加密流程图

2) KeyStoreManagerService 使用 token 获得在 ActivityManagerService 中的组件信息。

3) 根据 ActivityManagerService 返回的组件信息获得 KeyStoreContentProvider 中组件所存储的密钥信息。

4) 在 KeyStoreManagerService 中使用返回私钥对数据进行加密，将数据通过 Binder 传递回 Intent 方法中，由 Intent 负责加密数据的填充。

4.2. 基于权限的数据传输解密策略

根据以上返回的加密数据，在真正返回给客户端之前需要进行判断决定是否显示给用户。这里以 intent 传输为例，在 get 数据之前，被访问者返回了加密数据以及唯一标识，例如：intent.getStringExtra (“_permission”)，该返回的字符串数据是唯一标识，格式为：“包名+/+组件名+:+权限”，该数据返回值则确认返回的信息是否是经过加密后的数据，访问者需要通过验证才能解密真正获取数据。利用字符串分割，获得 permission 值，传递组件 token 到 ActivityManagerService，然后在 ActivityManagerService 中获得组件信息，并会调用 Android Framework 层的方法 PackageManager.getPackageInfo (packageName, PackageManager.GET_PERMISSIONS)获取权限，并检测这些权限是否是保护唯一标识中包含的权限，如果是恶意攻击者通过特权提升获取此数据，则本身是没有获取敏感数据的权限，此时则返回异常，不进行解密，抛出异常。如果存在，则根据此唯一标识提取数据库中的主密钥，并获取工作密钥，调用 KeyStoreManagerService 中的方法 decrypt 对加密数据进行解密，最后把明文数据放回给客户端显示出来。此时解密过程结束。解密过程如图 4。

1) 在 Intent 的获得数据的方法中，调用 KeyStoreManagerService 的 decrypt 方法，并将调用解密方法的所在组件的 Activity 对象的 token 和 intent 的_permission 数据传递到 decrypt 方法中。

2) 在 KeyStoreManagerService 的 decrypt 中，使用 token 在 ActivityManagerService 中获得该 token 所表示的组件的信息，并判断是否具有_permission 对应的权限，如果没有，则返回异常。

3) 如果该组件有对应权限，通过_permission 获得 KeyStoreContentProvider 中密钥信息，并使用私钥进行解密(如果组件权限伪造，那么获得的密钥也不会是真的密钥，解密后的数据也无法查看。)，将解密后的数据通过 Binder 传递回调用 decrypt 的 Intent 中，由 Intent 负责解密后数据的处理和返回。

5. 结论

本文通过对现有的安全进行了分析，并提出了修改框架层增加加密服务的设计方案，组件传输的数据加密可以一定程度的防范特权提升攻击，保证数据的安全性，此套框架也就可以直接被使用，但是也存在一些不足，由于有加密解密的一系列操作，其效率还需进一步提高，从而提高用户体验。

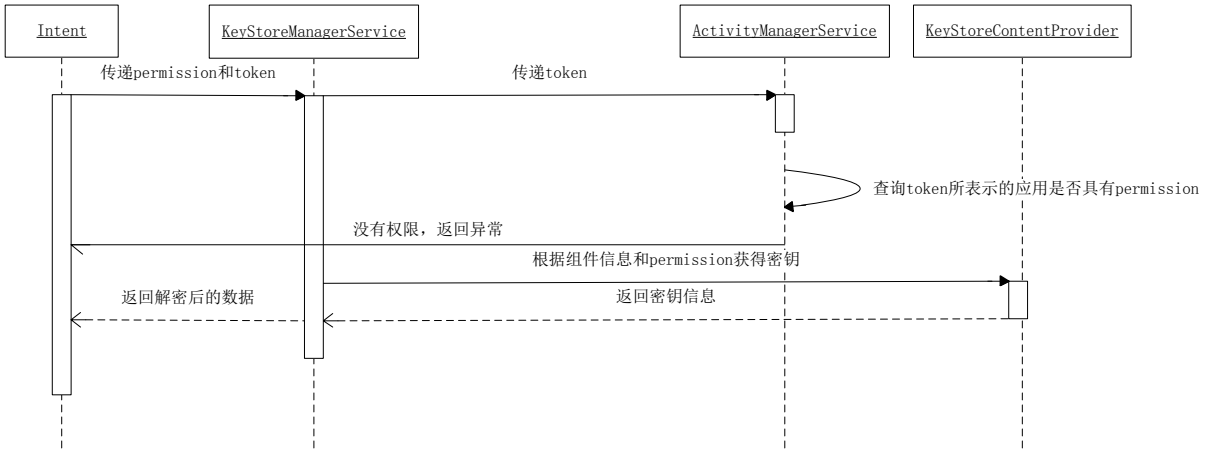

Figure 4. The flow chart of decryption
图 4. 解密流程图

基金项目

国家自然科学基金项目(61070207, 61370195)，北京市自然科学基金项目(4132060)和“十二五”国家密码发展基金密码理论课题项目(MMJJ201201002)。

参考文献 (References)

[1] Octeau, D., Mcdaniel, P., Jha, S., Bartel, A., Bodden, E., *et al.* (2013) Effective Inter-Component Communication Mapping in Android with *Epicc*: An Essential Step towards Holistic Security Analysis. *Proceedings of the* 22*nd USENIX Security Symposium*, Washington DC, August 2013, 543-558.

[2] Cui, X.M., Yu, D., Chan, P., Hui Lucas, C.K., Yiu, S.M. and Qing, S.H. (2014) CoChecker: Detecting Capability and Sensitive Data Leaks from Component Chains in Android. *Proceedings of the* 19*th Australasian Conference on Information Security and Privacy* (*ACISP* 2014), Springer-Verlag, 446-453.

[3] Backes, M., Bugiel, S. and Gerling, S. (2014) Scippa: System-Centric IPC Provenance on Android. In: 30th Annual Computer Security Applications Conference. http://dx.doi.org/10.1145/2664243.2664264

[4] Nauman, M., Khan, S., Othman, A.T., *et al.* (2014) Realization of a User-Centric, Privacy Preserving Permission Framework for Android. *Security & Communication Networks*, **8**, 368-382. http://dx.doi.org/10.1002/sec.986

[5] Bugiel, S., Davi, L., Dmitrienko, A., *et al.* (2011) XManDroid: A New Android Evolution to Mitigate Privilege Escalation Attacks. Technische Universat Darmstadt Center for Advanced Security Research, Darmstadt, 4-6.

[6] Beresford, A.R., Rice, A., Skehin, N. and Sohan, R. (2011) MockDroid: Trading Privacy for Application Functionality on Smartphones. *Proceedings of the* 12*th Workshop on Mobile Computing Systems and Applications* (*Hot Mobile*11), ACM, 49- 54.

[7] Hornyack, P., Han, S., Jung, J., Schechter, S. and Wetherall, D. (2011) These Aren’t the Droids You’re Looking for: Retrofitting Android to Protect Data from Imperious Applications. *Proceedings of the* 18*th ACM Conference on Computer and Communications Security* (*CCS*22011), Chicago, 639-652.

[8] Davi, L., Dmitrienko, A., Sadeghi, A.-R. and Winandy, M. (2010) Privilege Escalation Attacks on Android. *Information Security-International Conference*, **6531**, 346-360.

[9] Bugiel, S., Davi, L., Dmitrienko, A., Fischer, T., Sadeghi, A.-R. and Shastry, B. 2012 () Towards Taming Privilege-Escalation Attacks on Android. *Proceedings of Annual Network & Distributed System Security Symposium*, **130**, 346-360.