基于 Android 平台的文档保护方案设计与实现

邓莅川[1,2,3] 周 健[1,3] 雷灵光[1,3] 向 继[1,3]

[1](中国科学院信息工程研究所信息安全国家重点实验室 北京 100093)

[2](中国科学院大学 北京 100049)

[3](中国科学院数据与通信保护研究教育中心 北京 100093)

摘 要 近年来,随着 Android 平台在移动设备上的普及以及企业移动办公快速发展的趋势,Android 平台上隐私文档的保护显得越发重要。基于已有文档保护方案的研究,提出一套可供普通应用程序使用的轻量级的移动终端文档保护方案。该方案使用多重密钥技术并结合密钥拆分算法,在保证文档安全性的同时,实现了对用户透明的文档加解密。该方案还可对受保护文档进行实时监控,以保证其在整个生命周期内的安全性。基于该方案在 Android 平台上实现了一个原型系统,并在多个 Android 手机平台上进行了测试,实验结果验证了该方案的可行性和兼容性。

关键词 文档保护 Android 密钥拆分 文档监控 移动办公

中图分类号 TP3 **文献标识码** A DOI:10.3969/j.issn.1000-386x.2017.02.057

DESIGN AND IMPLEMENTATION OF DOCUMENT PROTECTION SCHEME BASED ON ANDROID PLATFORM

Deng Lichuan[1,2,3] Zhou Jian[1,3] Lei Lingguang[1,3] Xiang Ji[1,3]

[1](*State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China*)

[2](*University of Chinese Academy of Sciences, Beijing 100049, China*)

[3](*Data Assurance and Communication Security Research Center, Chinese Academy of Sciences, Beijing 100093, China*)

Abstract As mobile devices of Android are becoming more and more prevalent among enterprises as well as the individuals, the protection of private documents on Android platform is getting more and more important. On the basis of existing researches, a lightweight mobile document protection scheme for general applications is proposed, which used the technology of key splitting and multiple-key system. The process of encryption and decryption is transparent to the user without any perceptible delay. The scheme provided real-time monitoring to protect the confidentiality of documents during the whole lifecycle. Moreover, a prototype system on Android platform has been implemented based on the scheme, and it had been tested on several Android platforms. Experiment results show that the scheme is practical and compatible with the mainstream Android platforms.

Keywords Document protection Android Key split Document monitor Mobile office

0 引 言

随着移动通信技术的发展,移动办公已成为大势所趋。对于企业移动管控而言,移动办公在提供便捷服务的同时,也带来了一些安全隐患。层出不穷的手机病毒、木马等使得手机环境不安全,文献[1]表明中国企业普遍缺乏对电子文档的保护措施;而 SearchSecurity 网站调查显示,30% ~40% 企业机密泄露事件是由电子文档泄漏造成[2]。Android 系统自身的安全性同样存在不足,如 Android 手机中的 APP 安装缺乏有效监管,黑客可通过将恶意代码注入正常 APP 中来窃

收稿日期:2015 - 12 - 14。国家高技术研究发展计划项目(2013AA01A214)。**邓莅川**,硕士生,主研领域:移动终端安全。**周健**,助理研究员。**雷灵光**,助理研究员。**向继**,高级工程师。

取用户私密信息。近年来,通过远程控制窃取用户文档的事件时有发生[6]。

不同于 iOS 采取整盘加密的机制,Android 文件系统的安全性主要依赖于 Linux 文件系统基于用户身份的访问控制保护机制,即不同的用户组对文件有不同的读写权限。然而,Linux 内核存在一些安全漏洞[13],一旦攻击者利用这些漏洞获取了系统的 Root 权限,则整个 Linux 文件系统都会暴露给恶意代码,任何存储位置都是不安全的。随着普遍增加的企业文档共享需求,基于 Android 平台移动办公的数据共享安全性也得到越来越多的关注。因此,如何实现 Android 平台上隐私文档的保护显得越发重要。

国际上不少大学以及研究机构都在电子文档安全管理研究方面取得了一些成果,如麻省理工学院(MIT)、IBM 研究院等均各自提出了相应的理论及系统模型[7]。国内的清华大学、南京大学等高校学者也在电子文档加密、数据分发、密钥管理及访问控制等方面做了深入研究[8-9];北京邮电大学的学者提出将可信计算引入到移动平台中[14],实现对文档的透明加解密,其安全性依赖于一个关键安全函数,但在对关键安全模块的保护及性能开销问题上还有待进一步研究。

将传统 PC 文档保护技术应用到移动终端,需考虑手机有限的计算及存储能力,以及密钥在 Android 平台上存储时缺乏信任根等问题。手机文档保护需保证文档存储、传输和使用三个环节的安全性,其中对文档数据传输和文档载体进行管控是使用最为广泛的措施[3]。目前手机文档保护方案主要有以下几种:一是不在客户端存储文档,此时文档的安全性依赖于客户端和服务器端的安全传输以及服务器端的安全存储[10],但这种方案并不能有效阻止本地恶意程序的攻击;二是切断其他数据传输通道,如禁止用户在操作文档时使用网络服务[3],该方案同样无法阻止本地恶意程序攻击;三是改进文档载体,如提供企业专用的文档阅读器来禁止一些不安全操作[12],但阅读器的定制开发量大,且兼容性较差,还会在一定程度上影响用户的操作习惯;四是对文档进行单纯加解密[11],在该方案中解密后的新建文件将被放入 SD 卡,缺乏有效的保护控制。因此,寻求一种能有效解决上述问题的手机文档保护机制十分必要。

综上所述,本文针对 SD 卡丢失问题及手机后台恶意程序窃取机密文件的攻击,提出了一套移动终端文档保护机制;并以 SDK 的形式进行实现,方便第三方应用程序调用,具有良好的兼容性。本方案对需要保护的 SD 卡文档进行多因素密钥加密保护,且不在本地存储密钥,也无需密钥传输操作,因而保证了密钥的安全性;本方案将对解密后的临时明文文档进行全生命周期监控及进程间加锁,确保临时明文文档的安全创建、访问和销毁,且仅能由指定程序访问,还可以按需设置自定义监控策略;方案以 SDK 形式实现,方便应用程序使用,且无需对阅读器进行任何修改,满足了文档共享的需求。实验测试表明,本方案开销较小,且支持对目前通用文档格式的保护。

1 背景知识

Android 平台主要提供以下五种文档存储和共享的方法:

- SharedPreference

它是 Android 提供用于存储简单配置信息的机制,采用 Map 数据结构来存储数据,以键值的方式存储。只能在应用程序内部使用。SharedPreferences 以 XML 格式将数据存储到设备中,文件系统中的位置在/data/data/ < package name > /shares_prefs 下。

- SQLite 数据库

需对存储数据进行较复杂操作,或数据量较大时,需要采用 SQLite 来更高效管理数据,它是一种独立的无需服务进程,支持事务处理,可以使用 SQL 语言的数据库。

- ContentProvider

即应用程序间共享数据的统一接口,只有拥有相应权限才可获得 Content provider,并查询它们包含的数据。

- 文件存储(内部存储)

系统提供 openFileInput( ) 和 openFileOutput( ) 方法来读取设备上的文件,且只能在应用程序内部访问。文件系统中存储路径一般为:/data/data/PackageName/files。文件存储有四种操作模式:MODE_PRIVATE、MODE_ APPEND、MODE_WORLD_READABLE、MODE_WORLD_WRITEABLE

- SD 卡文件(外部存储)

Android 访问外部存储即 SD 卡文件需要“WRITE_EXTERNAL_STORAGE”权限,并需在使用前检查外部存储的可用性。

以上五种文档存储和共享方式中,内部存储安全性较高,但共享性较低;外部存储只需有相应权限即可访问,相对安全性低,但共享性更高。为兼顾文档安全性和共享性需求,本文将重点关注如何提高 SD 卡文

件存储的安全性。

2　方案设计

由于智能移动终端有文档保护和文档共享的双重需求,而现有的移动终端文档保护方案很少能同时兼顾这两点需求。因此,本文提出一套文档保护方案,以期实现安全的文档共享。本设计方案架构基于 Android 平台,并以 SDK 形式实现,方便集成到第三方应用程序中,仅需用户在程序初始化时提供用于身份认证和透明加解密的 PIN 码,具有很好的普适性。本方案可以提供集中管理文档的功能,适用于企业移动办公。

2.1　攻击模型

对于存储在 SD 卡上的文档,本文考虑以下几种攻击:

- 攻击者通过窃取 SD 卡来获取 SD 卡上文件;
- 恶意程序后台访问 SD 卡文件;
- 在手机 root 的情况下,恶意程序后台访问应用本地存储。

本方案针对上述攻击提出相应的防护策略,来自内部的信息泄露不在本文讨论范围。

2.2　保护模型

针对 2.1 节所述攻击模型,本文提出保护模型如图 1 所示。

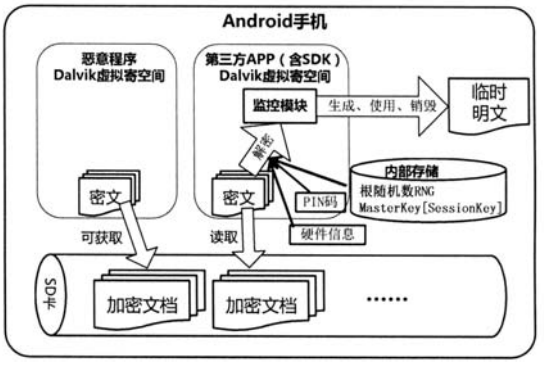

图 1　保护模型图

受保护文档在 SD 卡上加密存储,恶意程序后台访问 SD 卡文件时,由于没有密钥信息故无法破解密文。密文文档当且仅当用户通过 APP 进行显式请求时,才会被解密,且密钥只部分存储在本地受保护区域,故敌手即使通过 root 拿到部分密钥,也无法完成解密工作。

恶意程序在自己的 dalvik 虚拟空间内,无法访问第三方 APP 的虚拟空间,故也只可获取 SD 卡上的加密文档。第三方 APP 接收到打开文档的请求时,由用户选择阅读器,进而申请阅读文档,文档将在第三方 APP 的虚拟空间内被解密放到临时空间中,由文档监控模块监控保护;阅读器操作完成后即刻销毁明文并回收临时空间,该过程中恶意程序无法获取文档明文。

该保护方案相当于为 APP 提供了一块私密且安全的“保险箱”,对文档的加解密以及对明文空间的监控都与 APP 在同一个 dalvik 虚拟机[4]里。

(1) 密钥机制

文档安全性很大程度依赖于密钥保护,因此,需设计一套完整、安全的密钥生成、存储、使用和管理机制。

本方案采用密钥拆分和多重密钥思想,主要涉及三种密钥:用于加密文档的密钥 SessionKey、用于加密 SessionKey 的主密钥 MasterKey、用于加密 MasterKey 的根密钥 KeyKey(只在程序启动时动态获取);以及用于身份认证的 PIN 码。上述这些敏感数据都不在本地存储,也无需传输,故安全性得到了保证。

由于本方案的实现均在客户端进行,故采用对称加密算法。

为了提高移动办公的效率,本方案考虑把用于加解密文档的密钥暂存于手机客户端,并引入 MasterKey 来保护文档密钥 SessionKey,以杜绝密钥泄露的风险;本方案中客户端仅维护一个映射表:文档唯一识别号和 MasterKey [SessionKey] 加密后字符串一一对应。

符号表示如下: PIN 为用于用户验证的字符串;RNG 为根随机数,用于生成保护主密钥的密钥,由固定密钥加密存储于设备;INFO 为设备硬件信息,将密钥与设备绑定,确保更换设备时当前密钥失效,设备信息不存储,直接从系统中提取;random 为每个文档对应的随机数,不存储,只用于第一次加密时生成 SessionKey;SK 表示加密文档的密钥 SessionKey(SK’表示加密后);MK 表示加密 SessionKey 的密钥 MasterKey(MK’表示加密后);KK 表示加密 MasterKey 的根密钥 KeyKey;m 表示文档明文,s 表示文档密文;E(*)表示对称加密过程,D(*)表示对称解密过程;P(*)为既定密钥生成算法。

则　KK 生成:KK = P(PIN + RNG + INFO)
主密钥 MK 加密:MK’ = E(KK , MK)
主密钥 MK 解密:MK = D(MK’, KK)
SK 生成:SK = P(PIN + random + INFO)
文档密钥 SK 加密:SK’ = E(SK , MK)
文档密钥 SK 解密:SK = D(SK’, MK)
文档加密:s = E(m , SK)
文档解密:m = D(s , SK)

(2) 文档监控

文档保护方案一般涉及以下两种情况:一种是文

档解密后使用指定阅读器打开,从而保证其安全性,但这种方法兼容性和扩展性较差;另一种是将明文缓存,而缓存的明文易受攻击,安全性较低。

本文方案由监控模块监控针对文档的各项操作。方案中在操作文档的请求通过验证并由用户手动选择第三方阅读器后,才执行解密操作;文档明文被放在监控模块新建的临时空间中,并通过设置该阅读器打开文档时的参数,以确保其他阅读器对其无法访问;阅读器打开文档耗时很短,临时明文只存在很短的时间,故降低了本方案被攻击的可能性;在阅读器退出操作后,临时文件空间被程序自动回收且文档明文被即刻销毁。

2.3 模块设计

本方案以 SDK 形式实现,第三方 APP 可自行调用接口函数。SDK 主要由以下几个模块组成:

- 密钥管理模块:主要负责生成密钥,验证用户 PIN 码;
- 数据加解密模块:主要负责对文档加解密,及对随机数的加解密;
- 文档监控模块:提供对文档整个生命周期的监控保护,解密后临时明文存于新建临时空间,并对临时明文的操作进行实时监控,判断阅读器关闭文档后销毁明文。

模块如图 2 所示。

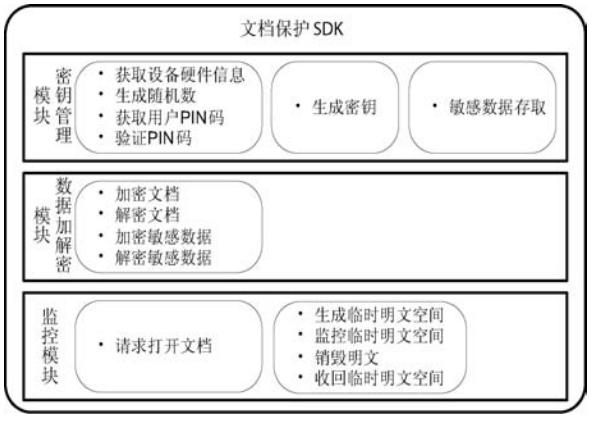

图 2 文档保护方案模块设计图

3 方案实现

由于 Android4.1 及之后的 4.2、4.3 统称为 Android Jelly Bean,包括后续的 Android4.4 是目前 Android 市场份额占有最大的系统版本。本节给出了一个基于 Android 平台 4.2 及以上版本的手机文档保护方案实现原型,分为初始化、加密和解密进行流程分析,最后进行相应的技术和性能分析。

3.1 实现流程

(1) *初始化*

初始化流程如图 3 所示。

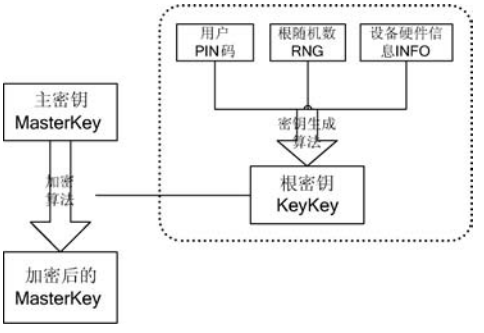

图 3 文档保护方案初始化流程图

- 生成设备硬件信息 INFO;
- 对指定信息 INFO 用指定算法进行一次加密,用于后续验证 PIN 码正确性;
- 生成一个“根随机数”(RNG),存于私有空间;
- [PIN + RNG + INFO]用指定算法生成根密钥 KeyKey;
- 生成任意字段,即与保护密钥无关的信息,作为主密钥 MasterKey;
- KeyKey[MasterKey]加密后存储于私有空间。

初始化过程仅在程序启动时执行,其余时候可以直接从内存读取 MasterKey。

(2) *加密*

加密流程如图 4 所示。

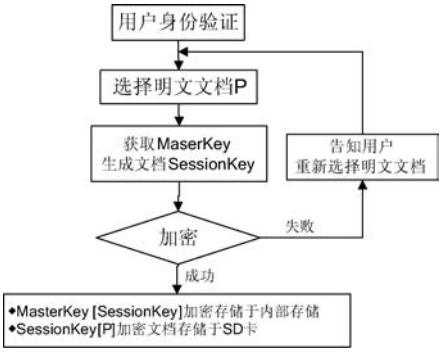

图 4 文档保护方案加密流程图

用户输入 PIN 码,解密预存储的 PIN[INFO]进行判断,解密成功则通过验证,程序获取 MasterKey;身份认证失败则提示用户重新输入 PIN 码,连续错误五次即被锁定,不能选择文档。用户通过验证后,在不退出程序的前提下可多次选择文档进行加密。

选定文档后,后台生成三个因素的字符串,即设备硬件信息、文档对应随机数、PIN 码,按指定算法生成密钥 SessionKey,用 SessionKey 加密文档。

加密成功则将 MasterKey [SessionKey]加密保护后存储到程序私有空间;加密失败则告知用户,并可重

新选择文档进行加密。

(3) 解密

解密流程如图 5 所示。

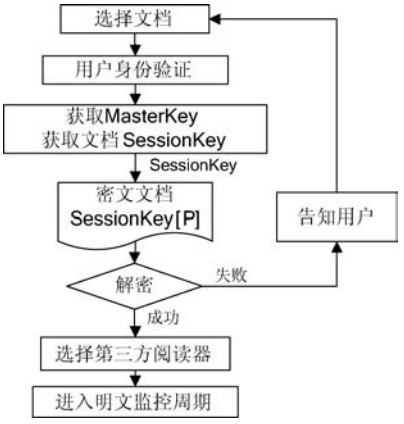

图 5　文档保护方案解密流程图

用户选择想阅读的文档,程序需用户输入 PIN 码,程序按(2)所述方法验证 PIN 码。

正确则进而判断内存是否有 MasterKey,有则用 MasterKey 解密得到 SessionKey 对文档进行解密;

内存没有 MasterKey,即程序启动初始化,后台获取三部分拆分信息,动态生成根密钥 KeyKey,解密得到 MasterKey。用 MasterKey 获取文档 SessionKey 进而对文档解密。

解密失败则告知用户,重新选择文档;解密成功则进入文档监控周期。文档监控周期处理流程如下:

① 文档创建:监控模块动态创建临时空间,将密文文档解密后暂时存放;

② 文档打开:通过获取 sun. nio. FileChannel 对象实现进程加锁,其他程序进程无法访问临时明文空间,从而其他的第三方无法获取到明文信息;

③ 文档关闭:阅读器对文档的阅读和操作都是在自己的 dalvik 虚拟空间中进行的,其他 APP 不能访问该虚拟空间也不能获知其是否操作完成,本方案利用 Android 系统提供的对 SD 卡监听机制,周期性地监听临时明文空间下对文档的操作,一旦发现临时明文文件在预置时间长度内处于空闲状态,即刻销毁明文。

考虑到大文档直接进行加解密处理可能效率很低,甚至可能出现内存溢出错误。本方案的文档加解密使用分块处理,则对于大小文件的处理在内存消耗上差异不大, 3.2 节的实验分析亦验证了这一点。

3.2　分析及实验

(1) 性能分析

- 安全性

本方案可以抵抗以下几种安全威胁:

a) SD 卡丢失,由于密钥需设备硬件号生成,SD 卡丢失无法获取密钥。

b) 恶意程序后台访问 SD 卡文件,没有用户 PIN 码;即使窃取得到用户 PIN 码,亦无法访问内部存储,因而无法获取完整密钥。

c) 恶意程序通过 root 获取内部存储的部分密钥,也无法获取完整密钥。

- 可用性

a) 加解密透明,无需改变用户使用习惯。

b) 本方案采用的算法对文档进行分块处理,文档加解密过程性能开销较小。

c) 无需定制阅读器,对通用阅读器均能支持。

- 算法性能分析

本文先用普通对称加密算法(AES)对不同大小的文件进行加解密,实验结果表明加解密开销与文件大小呈正相关关系,且大文件(如大于 30 MB)的加解密造成时间开销过大,会对用户操作造成不良影响。假设文件大小为 N,此时加解密算法复杂度为:$O(N)$ 。

在对大文件的处理方面,本方案做了如下优化,对文档进行预处理操作,分为大小为[1024 × 1024]位的块,再调用对称加密算法进行加解密,从而加解密的性能开销只与分块大小有关。下面将通过实验进行分析验证。

(2) 实验分析

本文针对方案的兼容性和性能开销进行了实验。兼容性即本方案支持通用的几种 Android 平台阅读器可用,支持通用的几种文档格式可加解密;性能开销即受本方案保护的 APP 对设备内存开销较小。

表 1 为普通对称加密算法、分块优化后的算法,以及不同分块大小造成的性能开销。

表 1　分块优化文档保护方案性能开销

格式	文件大小	电池功耗/mW				
		0	1	2	3	4
docx	20 KB	217	241	245	186	242
pdf	212 KB	240	246	274	225	220
pdf	1.92 MB	281	253	254	282	292
txt	4.76 MB	307	359	289	306	352
pptx	7.22 MB	281	422	312	366	269
pdf	35.8 MB	226	399	354	408	417
pdf	63.6 MB	299	352	262	415	388
格式	文件大小	内存用量/MB				
		0	1	2	3	4
docx	20 KB	5.97	6.16	6.16	6.11	5.93
pdf	212 KB	5.77	5.93	6.51	6.44	5.77
pdf	1.92 MB	6.12	13.86	9.17	6.95	6.14
txt	4.76 MB	6.36	11.57	8.87	6.95	6.01
pptx	7.22 MB	6.17	17.83	10.12	6.51	6.32

续表 1

格式	文件大小	内存用量/MB				
		0	1	2	3	4
pdf	35.8 MB	6.16	18.30	10.19	7.24	6.36
pdf	63.6 MB	6.17	18.07	9.16	6.95	6.34
格式	文件大小	时间延迟/s				
		0	1	2	3	4
docx	20 KB	3.75	4.80	4.35	4.52	4.61
pdf	212 KB	5.50	5.21	4.18	4.41	4.03
pdf	1.92 MB	14.40	10.51	9.26	10.32	10.56
txt	4.76 MB	9.01	5.52	6.85	4.6	5.46
pptx	7.22 MB	22.23	12.50	12.23	12.16	12.43
pdf	35.8 MB	50.10	27.63	29.25	29.01	28.94
pdf	63.6 MB	85.34	45.02	52.21	49.10	49.34

表 1 中 0、1、2、3、4 分别代表:未分块、分块大小为[2048 × 2048]、[1024 × 1024]、[512 × 512]、[256 × 256]。

表 1 内存和电池功耗均为本方案原型的开销,时间延迟为加密、存储、解密,及阅读器解析打开文件的总时间,其中文档中存在阅读器不能识别的内容,或媒体文件过多,会增加解析时间,引起更多的时间延迟;从时间延迟看出未分块的加密算法在处理大文件时延迟严重,而本方案对文档进行分块处理,从实验数据可以看出能够一定程度上减少时间延迟,且下一步还可加入多线程并行处理多个分块,将进一步优化算法性能,将时间复杂度降低至 $O(n)$,n 为分块大小;通用文档格式均能正常加解密,且不同格式文档加解密性能开销差异不大,打开文档的性能开销与阅读器处理识别不同格式文档有关。

表 2 为本方案原型在不同手机上对小文件(小于 1 MB)和大文件(大于 30 MB)进行加解密的性能开销。

表 2 文档保护方案在不同手机平台上性能开销

SDK 性能开销											
手机型号	文件大小/MB	系统版本	电池功耗/mW			内存用量/MB			CPU 占用率/%		
			加密	解密	空	加密	解密	空	加密	解密	空
Coolpad 7620L	<1	4.3	278	303	114	14.45	15.09	8.39	1.2	1.1	1.0
	>30	4.3	251	375	104	15.12	15.41	8.93	4.1	2.8	2.0
Huawei G750	<1	4.2.2	200	200	70	12.09	11.77	5.21	2.4	2.5	0.9
	>30	4.2.2	390	570	100	10.95	11.50	7.68	7.5	11.0	1.2
Sony LT29i	<1	4.1.2	892	757	157	15.89	14.99	6.61	1.6	1.7	0.19
	>30	4.1.2	618	636	128	20.30	20.91	7.20	4.2	3.7	1.2

表 2 中电池功耗为文档加解密时的手机当前功耗,可以看出在不同手机平台上加解密功耗均小于 1 W;手机在应用程序运行时的内存用量普遍保持在 20 MB 以下,且由于分块技术的使用,在对大小文档的加解密时内存用量差异不大;CPU 占用率普遍不超过 5%。以上实验在通用第三方文档阅读器,即 WPS Office、Polaris Office、Quick Office、ThinkFree Office 及金软 Office 上均进行了实验,均能正常使用本原型系统。实验表明本方案原型系统整体性能开销较小,且对大文件和小文件的加解密处理开销差异不大,故本方案可行。

4 结 语

手机文档安全对企业移动管理至关重要,仅靠现有的 Android 安全机制难以满足企业需求。本文设计开发了一套基于 Android 的轻量级手机文档保护方案,提供一套能满足企业移动办公需求的密钥管理体系,实现了文档的透明加解密功能,并兼容现有通用第三方阅读器方便通用文档的读取。本方案中的文档操作全程受后台监控,一旦操作完毕临时明文空间即被收回且文档明文被即刻销毁。经测试,本文实现的原型系统开销较小,具有较好的可用性和兼容性。在本方案的具体实现中,分组加密为串行处理,因此大文件可能带来一定的可感知延迟,实际应用中采用并行处理,将进一步降低时间复杂度,减少可感知延迟。

参 考 文 献

[1] 企业电子文档的安全调查报告[R/OL]. http://wenku.baidu.com/link? url = ga6q6 - NGJdMPhIUwJTNPm9OezAUn49t - kO3dtYG3S_otCWnQzMYi3BAxhtWBBt9LTxhxGzeuqB_zi2asjYmzTRS1_VNn0u2mREkt9GVApjO.

[2] 百度百科. 电子文档保护[DB/OL]. http://baike.baidu.com/view/1743385.htm.

[3] 施超. 信息安全的重要性与文档加密技术在企业中的应用[J]. 中国管理信息化,2015,18(6):203.

[4] Wikipedia. Dalvik (Software)[DB/OL]. http://en.wikipedia.org/wiki/Dalvik_(software).

(下转第 333 页)

nel rootkits with VMM-based memory shadowing[C]//11th International Symposium on Recent Advances in Intrusion Detection. Springer,2008:1-20.

[5] Hund R, Holz T, Freiling F C. Return-oriented rootkits: bypassing kernel code integrity protection mechanisms[C]// Proceedings of the 18th Conference on USENIX Security Symposium,2009:383-398.

[6] PAX. Homepage of The PaX Team[OL]. http://pax.grsecurity.net.

[7] 王曼丽,翟高寿. 基于编译器插件的轻量级内核重构加固方法研究[J]. 软件,2015,36(3):1-9.

[8] Chen S, Xu J, Sezer E C, et al. Non-control-data attacks are realistic threats[C]//Proceedings of the 14th Conference on USENIX Security Symposium. Berkeley, CA, USA: USENIX Association,2005:12.

[9] Garfinkel T, Rosenblum M. A virtual machine introspection based architecture for intrusion detection[C]//Proceedings of the 2003 Network and Distributed Systems Security Symposium,2003:191-206.

[10] Petroni N L, Fraser T, Molina J, et al. Copilot-a coprocessor-based kernel runtime integrity monitor[C]//Proceedings of the 13th Conference on USENIX Security Symposium, 2004:13.

[11] Chkrootkit: locally checks for signs of a rootkit[OL]. http://www.chkrootkit.org.

[12] RkHunter: protect your machine[OL]. http://www.rootkit.nl/projects/rootkit_ hunter.html.

[13] Petroni N L, Hicks M. Automated detection of persistent kernel control-flow attacks[C]//Proceedings of the 14th ACM Conference on Computer and Communications Security, 2007:103-115.

[14] Petroni N L, Fraser T, Walters A, et al. An architecture for specification-based detection of semantic integrity violations in kernel dynamic data[C]//Proceedings of the 15th Conference on USENIX Security Symposium,2006:289-304.

[15] Hofmann O S, Dunn A, Kim S, et al. Ensuring operating system kernel integrity with OSck[C]//Proceedings of the 16th International Conference on Architectural Support for Programming Languages and Operating Systems, 2011: 279-290.

[16] Riley R. A framework for prototyping and testing data-only rootkit attacks[J]. Computers & Security,2013,37:62-71.

[17] Baliga A, Kamat P, Iftode L. Lurking in the shadows: identifying systemic threats to kernel data[C]//Proceedings of the 2007 IEEE Symposium on Security and Privacy,2007:246-251.

(上接第 298 页)

[7] Shakiba M, Singh M J, Sundararajan E, et al. Extending Birthday Paradox Theory to Estimate the Number of Tags in RFID Systems[J]. PLoS One, 2014, 9(4):e95425.

[8] Yeh M K, Jiang J R. Parallel splitting for RFID tag anti-collision[J]. International Journal of Ad Hoc and Ubiquitous Computing, 2011, 8(4):249-260.

[9] 王雪, 钱志鸿, 胡正超, 等. 基于二叉树的 RFID 防碰撞算法的研究[J]. 通信学报, 2010, 31(6):49-57.

[10] 郭志涛, 程林林, 周艳聪, 等. 动态帧时隙 ALOHA 算法的改进[J]. 计算机应用研究, 2012,29(3):907-909.

[11] Zhang W, Guo Y, Tang X, et al. An efficient adaptivez anti-collision algorithm based on 4-ary pruning query tree[J]. International Journal of Distributed Sensor Networks, 2013, 2013:848746.

[12] 丁治国, 朱学永, 郭立, 等. 自适应多叉树防碰撞算法研究[J]. 自动化学报, 2010, 36(2):237-241.

[13] 王必胜, 张其善. 可并行识别的超高频 RFID 系统防碰撞性能研究[J]. 通信学报, 2009, 30(6):108-113.

[14] Shin J, Jeon B, Yang D. Multiple RFID tags identification with Mary query tree scheme[J]. IEEE Communications Letters, 2013,17(3):604-607.

[15] 吴博, 周铜, 王栋. RFID 防碰撞算法分析与研究[J]. 微电子学与计算机, 2009, 26(8):237-239,242.

(上接第 318 页)

[5] Android 系统各版本市场份额进化图[OL]. http://www.199it.com/archives/311622.html.

[6] 巧艳. Android 远程控制恶意软件兴起 可窃取用户信息[OL]. http://www.newhua.com/2013/0718/224074.shtml.

[7] Hennessy S D, Lauer G D, Zunic N, et al. Data-centric security: Intergrating data privacy and data security[J]. IBM Journal of Research and Development,2009,53(2):208-224.

[8] 郑磊,马兆丰,顾明. 基于文件系统过滤驱动的安全增强型加密系统技术研究[J]. 小型微型计算机系统,2007,28(7):1181-1184.

[9] 刘岸,吴琨,仲海骏,等. 基于策略机制的分布式文件保护系统 PFICS[J]. 计算机工程,2004,30(18):119-121.

[10] 廉喆. 手机文档保护系统的设计与实现[D]. 北京邮电大学,2010.

[11] 周巧扣,倪红军. 基于 Android 的文件加密系统的设计与实现[J]. 计算机光盘软件与应用,2013(16):245-246,248.

[12] 朱筱赟,胡爱群,邢月秀,等. 基于 Android 平台的移动办公安全方案综述[J]. 信息网络安全,2015(1):76-83.

[13] CVE 2009-1185[OL]. https://launchpad.net/bugs/cve/2009-1185.

[14] Yu X, Wen Q, Yan T. A novel solution to document protection on mobile platform[M]//Future Wireless Networks and Information Systems. Springer,2012:447-455.