

doi:10.3969/j.issn.1007-7162.2013.03.009

基于远程服务的 Android 平台下 SQLite 数据库安全机制的研究与实现

吴一民,何嘉荣
(华南理工大学 计算机科学与工程学院,广东 广州 510006)

摘要: 基于客户端/服务器的模型,对 Android 操作系统下的 SQLite 数据库提出了一个安全保障机制. 客户端除了基于 Blowfish 算法实现了 SQLite 数据库的加解密接口以外,还实现了一个本地服务器和一个身份认证应用程序. 远程服务器基于 J2EE 框架技术实现,为客户端提供密钥服务、挂失服务和身份认证服务.
关键词: Android 操作系统;SQLite 数据库;Blowfish 算法;加密;远程服务
中图分类号: TP309 **文献标志码:** A **文章编号:** 1007-7162(2013)03-0049-04

The Design and Implementation of the SQLite Security Mechanism Based on Android and Remote Service

Wu Yi-min, He Jia-rong
(School of Computer Science & Engineering, South China University of Technology, Guangzhou 510006, China)

Abstract: The security mechanism protects the SQLite database and is based on the client/server model. As for the client, the security mechanism not only fulfils the encryption and decryption interface functions with Blowfish, but also builds a local server. With regard to the server, the security mechanism provides key service for the client, loss-report service, and identification service.
Key words: Android; SQLite; Blowfish; encryption; remote service

随着移动终端技术的迅速发展,智能手机在人们的生活中扮演着越来越重要的角色.而智能手机给人们带来便利的同时,安全问题也进入了“危险”的境地.目前智能手机最主要面临的安全问题来自于手机系统^[1-3].

Android 操作系统中的绝大多数应用程序都会使用到数据库.但是,作为 Android 操作系统底层依赖的数据库,SQLite^[4]数据库竟然没有采用任何安全机制.所以,SQLite 数据库成为了 Android 操作系统一个很大的安全隐患.本文正是通过研究 SQLite 数据库的安全漏洞,提出一种基于远程服务的安全机制,通过保护 SQLite 数据库来提高 Android 操作系统的安全性^[5-6].

1 设计

1.1 安全机制概述

很多数据库系统,如 Oracle 数据库等,都引入了

安全保障机制.常用安全机制包括:用户认证、访问控制、数据加密和数据库操作审计等^[7-9].

用户认证:数据库系统通过鉴别用户的有效证明.只有验证通过,用户才能读写数据库.

存取控制:数据库系统为不同用户分配不同的权限.只有特定权限的用户才能完成特定的存取操作.

数据库加密:对数据库中的数据进行加密.

操作审计:对用户数据库中的所有操作进行记录,并作日后分析.

其中,数据库加密是较为有效的安全措施.因为攻击者可以通过数据库漏洞或者物理接触设备来绕过用户认证和存取控制直接修改数据库文件的内容,而数据库加密能够有效地防范这类威胁.但是,常用的数据库加密措施不适合 Android 设备中 SQLite 数据库的使用,部分常用数据库加密措施需要用户记住密钥.但是,绝大多数的 Android 应用程

序都有独立的数据库,若所有的数据库都采用同一组密钥,虽然能够减轻用户的负担,但是数据库的安全性将大大降低;若所有的数据库都采用独立的一组密钥,安全性虽然提高了但是用户的负担也增大了.又如部分数据库加密措施虽然负责密钥的生成和记录,但是密钥和数据库文件存放在同一个设备上会造成安全性较低等问题.

本文提出的安全机制目标是保证用户在 Android 客户端上的 SQLite 数据库的安全,以及在设备丢失以后能够最大限度地减少用户的损失.然而,

SQLite 数据库本身安全性较低,仅通过 Linux 文件系统的文件权限控制来提供存取控制.本安全机制不仅通过借鉴上面提到的安全措施为 SQLite 数据库增加用户认证和操作审计^[10-12],而且改进了数据库常用的加密措施并引入了挂失机制.

1.2 架构

图 1 为整个安全机制的架构图.本安全机制的设计是基于客户端/服务器模式,整个安全机制由身份认证模块、安全控制模块、通讯接口模块、加密模块、审计模块、密钥服务模块、用户账户功能模块组成.

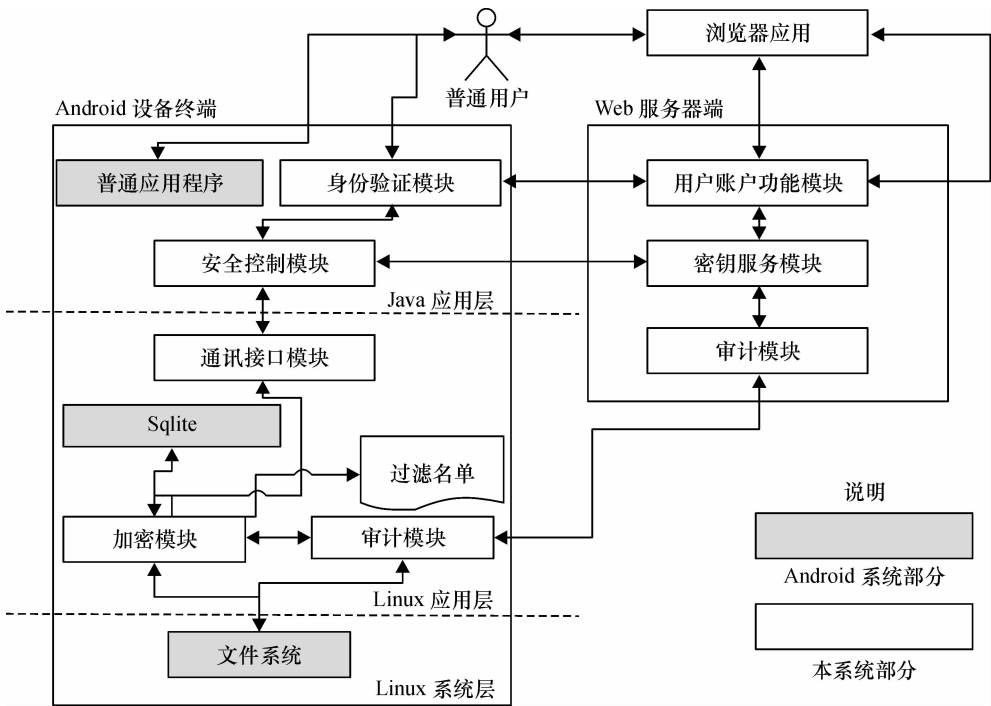


图 1 系统架构
Fig. 1 The architecture of the system

1.3 安全机制工作流程

在正常情况下,用户首先在 Android 客户端中通过身份认证模块进行身份认证.认证通过后,用户可以透明地操作 SQLite 数据库.同时,审计模块负责登记工作,加密模块负责完成期间的数据加解密工作.

在用户丢失移动设备的情况下,用户可以通过浏览器登陆到远程服务器中,并且通过用户账户功能模块完成挂失工作.那么,丢失的移动设备将接受到挂失信号并自动重启.若用户重新拾获该移动设备,可以再次登录到远程服务器的用户账户功能模块中完成取消挂失操作,那么用户就可以正常操作数据库.若移动设备被外人拾获,外人虽然能够接触到移动设备上的数据库文件,但是因为数据库文件已经加密,所以外人也不能查看或者修改数据库内容.

1.4 数据库文件密钥生成

本文采用 Blowfish 算法来加解密. SQLite 数据库

文件以大小固定的块状物理结构进行存取,Blowfish 算法使密文和明文长度一致从而有利于数据库文件加解密过程中的存取.另外,每个数据库文件都分配独立的密钥,这使得任意一个密钥的丢失都不会影响到其他数据库的安全性.远程服务器在成功接收到 Android 客户端发出创建数据库请求后,自动生成一个随机字符串作为该数据库的密钥. Blowfish 算法的密钥可以是 1~448 位二进制位.综合考虑安全性和性能,这里选取了 128 位.

1.5 数据库的加解密

当加密模块接受到远程服务器发送过来的密钥后,在 SQLite 数据库应用程序中实现的接口会将该密钥和 Blowfish 算法^[13]的实现函数嵌入数据库.然后,当 SQLite 数据库应用程序需要将缓存写到文件系统中的文件时,SQLite 数据库应用程序就会调用嵌入的 Blowfish 算法实现函数和密钥将数据通过加

密的方式写到文件. 当 SQLite 数据库应用程序需要读取数据库文件的时候, 就会调用嵌入的 Blowfish 算法实现函数和密钥将数据库文件中的密文进行解密并且写到内存中.

1.6 数据传输的安全性

Android 客户端需要不断地与远程服务器通信来保证操作的安全性. Android 客户端与远程服务器建立连接, 并发送和接收数据, 但使用明文数据进行传输是不安全的, 随时有被截获的可能. 安全机制通过迪菲-赫尔曼的密钥交换算法在远程服务器和客户端两边分别生成一个相同的密钥, 然后通过这个传输密钥加解密传输数据来确保数据交换的安全性^[14].

迪菲-赫尔曼算法是安全的密钥交换算法, 它的安全性建立在有限域内计算离散对数的复杂度上. 对于一个大素数, 想要计算出离散对数是几乎不可能的, 或者需要付出极大的代价.

每次通过迪菲-赫尔曼生成密钥后, 远程服务器和 Android 客户端操作系统都会各自存储传输密钥. 每当客户端向服务器发送请求信息时, 本安全机制会先在本地获取传输密钥加密请求信息. 而在另外一边, 远程服务器在接收到加密的数据后会先从数据库中找到传输密钥, 再对其解密.

1.7 挂失

Android 移动设备的移动性导致了它容易丢失或被偷窃. 当用户的 Android 移动设备丢失后, 用户通过浏览器登陆到远程服务器中, 并且通过用户账户管理模块完成挂失操作. 用户挂失设备后, Android 设备会接收到一个挂失信号并且实现自动重启. 同时, 远程服务器将拒绝处理该设备的一切请求. 若用户重新拾获该设备, 用户又可以通过浏览器登陆到远程服务器中, 并且使用用户账户管理模块提供的挂失恢复功能. 之后, 用户可以正常地操作设备中的 SQLite 数据库.

2 实现

2.1 加解密接口实现

在客户端中, 最核心的实现部分是 SQLite 数据库向外提供的加解密接口的实现. 其中需要实现的接口函数包括: `SQLite3CodecAttach()`、`SQLite3_key()` 和 `SQLite3_rekey()`.

(1) `SQLite3CodecAttach()` 函数负责将密钥嵌入数据库块. 若有提供密钥则创建结构体 `LPCryptBlock` 并将密钥嵌入其中, 然后将 `LPCryptBlock` 嵌入数据库块中; 若没有提供密钥, 则直接从数据库块中

检索 `LPCryptBlock`. 其中, 结构体 `LPCryptBlock` 包含密钥和解密函数.

(2) `SQLite3_key()` 函数负责将密钥嵌入数据库中. 首先, `SQLite3_key()` 函数会查找 `LPCryptBlock`. 若存在, 则直接返回; 若不存在, 则将传入的密钥重新封装成符合规格的密钥并且将其嵌入新创建的 `LPCryptBlock` 中, 最后将 `LPCryptBlock` 嵌入数据库.

(3) `SQLite3_rekey()` 函数负责修改密钥. 首先, `SQLite3_rekey()` 函数会检索 `LPCryptBlock`. 若数据库已加密, 则先修改写密钥并开启事务使用原来的读密钥读取密文, 然后用新的写密钥写回密文, 最后事务成功则返回, 否则回滚; 若数据未加密, 则按照 `SQLite3_key()` 函数的处理逻辑处理. 如图 2 所示.

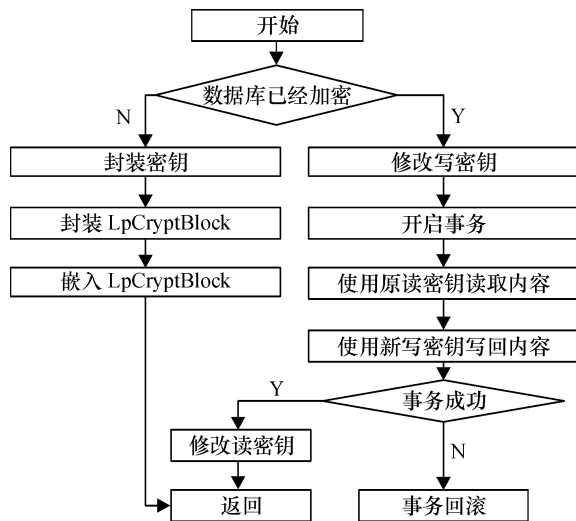


图2 SQLite3_rekey() 执行逻辑图

Fig. 2 The performance logic of SQLite3_rekey()

2.2 远程服务器实现

远程服务器负责: 处理 Android 客户端的各种请求, 包括用户身份验证、获取数据库加密密钥、记录日志等; 存储用户数据、数据库加密密钥、审计记录等各种数据; 提供用户完成注册、登录、挂失以及查看日志的界面. 基于以上设计, 远程服务器划分为下面 3 大模块实现: 用户账户模块、密钥服务模块和审计模块. 远程服务器采用 Spring 开源框架实现, 这里不作介绍.

3 功能测试

安全机制在安装了 Android 4.0.3 版操作系统的友善之臂 Tiny210 V2 开发板和部署了 Tomcat Web 服务器的远程服务器中进行功能测试.

未经过本安全机制加密的数据库文件通过 Linux 下的 `cat` 命令能够直接查看其内容. 而经过本

安全机制加密的数据库文件通过 Linux 下 cat 命令打开将显示为乱码. 由此可见, 本安全机制能够对数据加密.

用户在使用本安全机制的过程中, 必须在客户端中通过登录界面与远程服务器通信并且完成身份认证才能正常使用 SQLite 数据库. 用户验证通过后, 客户端中的应用程序就可以透明地操作 SQLite 数据库, 如通讯录应用程序. 由此可见, 本安全机制能够提供用户认证.

用户可以通过浏览器登陆到远程服务器中查看日志信息并且执行挂失操作. 一旦用户执行挂失操作, 客户端就会马上自动重启. 客户端重启后, 通讯录应用程序会因无法从远程服务器中获取密钥而不能正常查看数据库中的数据, 如图 3 所示. 可见, 安全机制实现了操作审计和挂失功能.

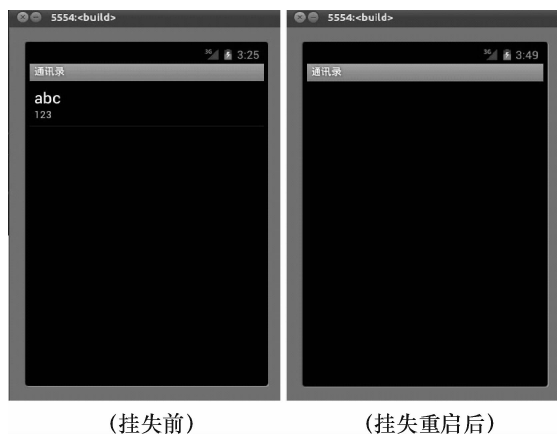


图3 挂失前后通讯录应用程序运行效果图

Fig.3 The running effect of the contact application before and after report-loss

4 结束语

本文提出的安全机制基于客户端/服务器模式并具有良好的安全性和易用性.

其安全性体现在: 实现了密钥与数据库文件存储的物理隔离; 每一个数据库都将分配独立的密钥, 所以任一密钥的丢失都不会影响另外一个数据库的安全和使用; 提供挂失功能.

其易用性体现在: 密钥管理和加解密过程对用户透明, 均由远程服务器与 Android 客户端协同完成.

当然, 本安全机制仍然存在一些不足, 需要在日后的研究中改进.

参考文献:

[1] Yang Tang, Phillip Ames, Sravan Bhamidipati, et al.

CleanOS: limiting mobile data exposure with idle eviction [C] // Amin M Vahdat. OSDI'12 Proceedings of the 10th USENIX conference on Operating Systems Design and Implementation. Berkeley: USENIX Association, 2012: 77-91.

[2] Roxana Geambasu, Steven David Gribble, Tadayoshi Kohno, et al. Keypad: an auditing file system for theft-prone devices [C] // Christoph Meyer Kirsch. EuroSys'11 Proceedings of the sixth conference on Computer systems. New York: ACM, 2011: 1-16.

[3] Mark D Corner, Brian David Noble. Zero-interaction authentication [C] // Ravi Jain. MobiCom'02 Proceedings of the 8th annual international conference on Mobile computing and networking. New York: ACM, 2012: 1-11.

[4] SQLite. About SQLite [EB/OL]. <http://www.sqlite.org/about.html>, 2013-3-20.

[5] Aubrey-Derrick Schmidt, Hans-Gunther Schmidt, Jan Clausen, et al. Enhancing security of Linux-based Android devices [EB/OL]. http://www.dai-labor.de/fileadmin/Files/Publikationen/Buchdatei/lk2008-android_security.pdf, 2008-9-17.

[6] Ross J Anderson, Markus Kuhn. Tamper resistance: a cautionary note [C] // USENIX. WOEC'96 Proceedings of the 2nd conference on Proceedings of the Second USENIX Workshop on Electronic Commerce. Berkeley: USENIX Association, 1996: 1.

[7] 吴涛峰, 张玉清. 数据库安全综述 [J]. 计算机工程, 2006, 32(12): 85-88.

Wu Tao-feng, Zhang Yu-qing. An overview of database security [J]. Computer Engineering, 2006, 32(12): 85-88.

[8] 臧劲松. 数据库系统安全的研究与分析 [J]. 计算机安全, 2008(7): 26-30.

Zang Jin-song. Research and analysis of the database system security [J]. Computer Security, 2008(7): 26-30.

[9] 王珊, 萨师煊. 数据库系统概论 [M]. 4 版. 北京: 高等教育出版社, 2006. 130-149.

[10] 廖顺和. 嵌入式数据库 SQLite 加密方法分析与研究 [J]. 计算机应用与软件, 2008, 25(11): 256-258.

Liao Shun-he. Analysis and research of the encryption method for SQLite [J]. Computer applications and software, 2008, 25(11): 256-258.

[11] 李凡. Android 系统安全机制的分析与增强 [D]. 武汉: 华中科技大学计算机科学与技术学院, 2012.

[12] 刘晖, 彭智勇. 数据库安全 [M]. 武汉: 武汉大学出版社, 2007.

[13] Bruce Schneier. The blowfish encryption algorithm-one year later [J]. Dr Dobbs's Journal, 1995, 20(9): 137.

[14] Douglas R Stinson. Cryptography: theory and practice [M]. 3rd ed. London: Chapman & Hall/CRC, 2005.