智能卡大作业报告

沈俊贤 计52 2015011258

周京汉 计52 2015011245

李睿燮 计55 2015080062

一、背景与问题分析

1、背景描述

MIFARE MF1是符合ISO/IEC 14443A的非接触智能卡。其通讯层（MIFARE RF 接口）符合ISO/IEC 14443A标准的第2和第3部分。其安全层支持域检验的CRYPTO1数据流加密。因为其安全性、可靠性和简易性，常常使用M1卡进行个人信息的存储。

2、重点难点

本实验的主要重点和难点在于：

- 如何形成一套完整的系统，并且让子系统之间的通信和协作能够顺利和流畅的运行。
- 如何形成用户友好的机制，因为这套系统最终的目标用户是可能完全不了解计算机的人群，所以如何让界面变得容易操作、让不规范操作变得能够容忍，也是非常重要的一件事情。
- 由于涉及到了需要在M1卡上存储相关的学生信息，而这是一个完全独立的信息存储单元，必须保证信息的安全性和不被篡改的要求，所以对安全方面的考虑就变得格外重要。

二、实验原理与方案

1、实验设计与问题解决

M1卡压力读取：

我们设计了以下的情景来进行M1卡压力读取的测试：

- 卡一直放在读卡器上的情况
    - 解决方案：由arduino部分的硬件模块检测，每次在loop循环开始的时候检测是否是新卡，如果不是新卡，不进行任何操作。
- 卡连续短暂拿开放回
    - 解决方案：主要的问题是会导致串口发送的信息重叠，无法分别是哪次读取的，为此我们引入了延迟，强制串口的信号在读取不同卡的时候相距0.5s。
- 卡在读/写一半的时候拿开
    - 解决方案：两侧的串口会监测串口中发送的内容，当卡移开的时候arduino会向客户端发送关闭信号，两侧都停止工作，等待下一次检测到卡的活动。
- 卡刚放上去就拿开

    - 解决方案：在实验中，这种情况包含在了上一种情况之中。
- 门禁系统要求能够连续读卡
    - 解决方案：门禁系统的连续读卡的要求令我们对一些可能产生错误的情况进行了宽松处理——如果发生错误不会退出程序，而是会发送门禁进入失败的信号，并且开始下一次检测。

M1卡内加密：

M1卡本身带有的安全机制是每扇区(四个块区)有一个trail block负责该扇区的读写权限. trail block的结构是keyA(6字节), access bits(4字节), keyB(6字节). 这个安全机制有两个问题:

- 如果access bits中写入无效值, M1卡的该扇区就不能访问, 相当于故障.
- M1卡的keyA,keyB只有6字节, 有暴力破解的危险. 目前的M1卡的漏洞研究说明M1卡还有克隆卡片, 重放攻击, 密钥窃听, 验证漏洞的危险. 因此不能只依靠M1卡安全机制.
  我们选择的方法是AES128. 16字节密钥. 16字节的输入和输出.
  恰好每个块区的容量也是16字节. 当前AES128在非量子计算环境下, 保证安全.

简单的设置密钥和用AES128加密还是不能避免克隆卡片攻击. 所以采用了M1卡固有的uid和特定的密码做异或得到密钥. 这样可以避免克隆卡片攻击.

听说还有一个M1卡叫万能卡可以修改其uid. 对于这个攻击, 有一个策略是让学生每次使用学生卡的时候, 输入密码, 然后再把输入的密码和特定的密码做运算得到密钥. 但考虑到学生卡的方便性, 放弃了这个策略.

2、实验实现：

M1卡读写：

在python应用中需要M1卡操作时, 在arduino完成此操作,然后 把返回值通过串口给python的. 依旧是python是发送命令, arduino是发送返回值. 分别是在tool.py和对应的ino文件中实现.

python发送到arduino的命令, 如下

- 读第i块区 -- "r i"
- 写第i块区 -- "w i 16字节数据"
- 清空 -- "clear"

arduino收到上面的命令后, 执行相应的操作并返回结果.

实现此功能的时候遇到过一个问题: 通过一次认证以后, 只能访问一个扇区. 不管别的扇区的密钥相等, 都不能访问. 第一次发现此bug时, 以为M1卡坏了, 试了别的M1卡, 还是不行. 最后发现不管扇区的密钥相等, 访问另一个扇区之前要更新认证.

底层读卡接口：

对于底层接口，我们采用了mfrc522的接口，主要包括读写某个块，认证，我们在地城读卡接口之上进行了封装和重用，让整个接口能够更加灵活的使用。

服务器：

我们我服务器是用django框架搭建的。在服务器端，我们主要是拥有两大功能：第一个是能够完成注册中心的功能；可以创建新的卡片信息以及可以为旧用户进行注册，即延长有效期；与注销卡片信息。第二个是服务器功能，可以用数据库存储所有用户的信息。

第一部分，在注册中心功能这里，我们应用了之前来自M1卡读写的接口，在调用之下可以直接对指定的block进行读和写的操作。在进行创建的时候，可以在服务器中访问`/testdb`，来进行创建和对于卡片的读写；在进行注销的时候，可以访问`/clear`来进行消除。

第二部分，在服务器中，每一次对应的读取卡操作，或者外面传来的对卡操作的请求，在通过判断之后都会对其中数据库的信息产生修改。在服务器的数据库之中，外面存储了一下信息：

- 姓名 -- name
- 学号 -- idnumber
- 院系 -- department
- 身份 -- identifier
- 性别 -- sex
- 有效期 -- validdate
- 余额 -- money

服务器应用了django原生的sqlite3数据库，代码结构使用了软工当中的分层结构，在最底层为中支持对数据库进行多种多样的读写操作；并且在上面一层增加了各种判定，保证输入的操作请求和返回的数据全部是合法，符合我们格式规定的数据，增加了安全性。再上面一层为将数据库与Arduino读卡接口连接的程序，最上面一层为支持url操作的代码。

另外，在安全方面，我们后面的设想是加入一个新的table表项在数据库之中，这个表项为由卡片的UID与系统中的16为key等信息进行异或操作得到的标识码，来增加对于卡的识别度，但是由于时间原因没有完成。

客户端：

客户端我们依然使用的是python3进行书写，其代码结构仍然使用了分层机构，包括和Arduino进行串口交流，调用交流接口，对客户端输入进行解析和对于接口的调用。

在客户端中每次启动的时候，系统都会自动向服务器端请求全部学生当前的信息，并存储起来，同时自动生成（或者）读取一份门禁信息。

在客户端，我们总共有三个模块：注册中心，零钱包，门禁。

首先注册中心这边的功能有：

- 创建新卡
- 延长旧卡有效期（注册）
- 获得卡片信息（丢卡做新的）
- 注销旧卡：分为只删除卡片和同时删除数据库中的信息
- 添加门禁信息
- 删除门禁信息

其功能完全覆盖了服务器端，甚至还多出来一些。其中在门禁这一块，我们的设定为：每次判断门禁，都会先读取卡内的基本信息，然后在本地进行学号判断，如果本地添加了该学号的信息，则可以进入下一环节；随后判断服务器状态，如果服务器可以连接，便联网询问服务器其信息时候有效，否则则在本地对其信息进行判断。

随后的是零钱包功能有：

- 查询余额
- 圈存
- 消费
- 查询消费记录

其为四个基本功能，在每次开启零钱包的时候就先读取其基本信息，获得余额和消费记录的显示。并且在每次花钱之后在本地更新余额等信息。

最后是门禁部分，本部分包括：

- 联网门禁
- 断网门禁

本部分由于界面并不是十分需要，因此没有写图形界面，是用shell进行启动和信息返回的。并且，其中的联网和断网是在每次进行门禁的时候进行判断的，只有在发现服务器不可用的时候才会进入相对不安全一点的断网门禁。

客户端界面：

客户端是python写的, 而python提供的GUI模块Tkinter是比较丑. 则选择了electron和zerorpc. electron是通过html, css, js实现桌面应用的工具. zerorpc是通过客户端和服务器的形式连接不同应用的工具. zerorpc一边在javascript可以用, 一边在python也可以用. 这样可以用html, css, js等完成前端. 通过zerorpc把前端和python连接. 提高实现前端的效率和美观性. 按照客户端的要求添加按钮,输入框,输出框就可以了.

三、实验结果与结论

1、客户端shell模式测试

由于其他的测试基本都在图形界面当中，客户端shell的主要测试部分为门禁。在实验展示的视频当中的5，6两部分可以看到这一部分的测试。测试结果十分流畅，可以自动判断服务器是否可用，并会正确的返回门禁是否通过。

其余部分的功能实际上也都可以通过shell进行操作，实际实验当中，由于没有Electron的通信延迟，shell操作速度反而更快，更加流畅！

2、客户端界面测试

客户端界面测试良好，可以完成全部功能，但是存在问题。在Electron与python的通信环节，会出现丢包现象，两边的信息可能会出现无法接收的情况，比如python返回的alert，主界面就会有时候无法收到，导致界面有时无法显示“操作成功”。

3、服务器创建注册

服务器可以流畅完成全部的注册功能，但是其操作种类有限，不能完全作为一个客户端进行操作。

4、读卡遇见的问题与解决

除了之前提到的M1卡压力读取问题，我们还遇到了M1卡读写中文的问题，因为中文无法用ascii码简单表示，同时utf-8的字节表示也不在0-128之间，同样无法用ascii码表示。而且utf-8一个汉字占用三个字节，不方便字节对齐，所以我们最后选用了unicode的方式存储，两个字节存储一个汉字。

四、思考与总结

李睿燮: 研究学生卡系统的安全机制时, 发现不能只依靠一个部分的安全机制. 比如, 对于一个既不能暴力破解既加密数据的绝对安全的M1卡, 只依靠卡本身的保护机制的话, 刷卡机是不能判断此卡是已挂失的卡还是正常的卡. 这种情况需要服务器的协助. 而且对于万能卡的克隆危险, 还需要依靠用户输入自己的密码. 这个让我联想到2018年初公布的meltdown漏洞, 类似地intel CPU漏洞导致要么重头改CPU结构或者要操作系统的协助. 因此, 在开发一个系统的安全机制, 在每个部分的开发者需要经常交流. 如果发现不能在自己负责的部分解决漏洞, 需要请求相应部分的开发者的协助.

沈俊贤: 我在做整个实验的过程中，一直认为这个项目的开发和软工项目的开发更加类似，因为M1卡的读写并不是一个特别大的难点，但是如何让整个系统更加易用则比这个更加难。我们在界面的设计、功能的实现等方面都考虑到了这些，同时通过这个实验，我也对平常习以为常的学生卡的设计产生了新的体会。

周京汉：这个实验是在小实验的基础上做的一个大拓展开发，在技术层面上其并不是一个很大的技术难关，我认为其难点更多的存在与如何利用一个更好的系统去能够流畅，安全的完成其所需要的全部功能。这即涉及硬件的问题（能流畅的完成其硬件功能），也涉及软件工程方面的知识（让整个系统鲁棒的运行起来）。这种综合考虑的需求让我们在完成大实验的过程中对多方面知识有了更加深刻的了解。