

# 作品报告

作品名称: 基于动态口令的 RFID Scanner 信息安全保障

电子邮箱: okcd00@qq.com / okcd00@gmail.com

提交日期: 2015.06.03

# 目 录

摘要 .....	1
第一章 作品概述 .....	3
第二章 作品设计与实现 .....	6
第三章 作品测试与分析 .....	13
第四章 创新性说明 .....	19
第五章 总结 .....	20
参考文献 .....	21

## 摘要

传感网是日益深入生活的物联网系统中不可或缺的重要部分，用以捕捉信息并传递给节点，采用多跳的形式，将采集到的数据信息反馈给服务器计算。自然而然地，我们对未来的智能家居产生了联想，各传感器协同为家中的服务器提供情报，从而达到自动门禁、智能水电、人物识别等功能……自然，安全隐患紧接而来：在实验中我们可以轻而易举地获得其他组的卡片信息或者节点传输的信息，倘若有不法之徒携带一套Scanner蹲守在住所外，那么家中的信息如同开了免提的电话一般暴露于外。不仅仅是智能家居，机场、气象研究机构甚至机场等公共设施的传感器，都由于节点耗能和存储空间局限，而不得已将信息传递过程暴露在公共的环境下。在此种情况下，既然节点的发展遇到了瓶颈，不如在其他方面考虑，就这样我们衍生出了想要为Scanner加密的想法。

于是现在这一套基于动态口令的RFID\_Scanner的雏形已初具规模。它的功能通俗一点说就是对“从原始信息到有效数据的转换”过程的保护，并初步形成一个具有预设值的自定义信息安全体系。基于动态口令这种定时更新的动态密码防护，相当于对试图盗取信息包破译的hacker规定了极其严苛甚至不可能完成的时间限，以此作为核心，在<Sensor-Scanner>（预设AES），<Scanner-Server>（预设MD5-Hash），<Server-Scanner>（预设AES）三个传输过程中，根据情境的不同（所需的安全等级/处理效率的要求不同），用户可自定义加密的方式以应对不同的需求（动态口令中所使用的MD5并非国有的Hash算法，为了更进一步的安全性，后续还将研究使用国有的Hash算法的SM3算法来进行替换）。

SafeScanner（项目名，以下简称为SS）的特性在于其动态性，自然也完全遵循信息安全的CIA三要素：Confidentiality保密性、Integrity完整性、Availability可用性。动态性体现于加密方式的多样性及可变性，由于破解需要在极短时间内完成，超过正常延时则完全无效化，以此大大加大了外来入侵者的破解难度与破解成本，动态的同时也一定程度上实现了保密性；自然，动态的内部保障机制之外，还有通用常用的AES、MD5散列表同时在外守护着数据信息的保密安全；此外，加密过程采取非截取式，加密/解密后保证原始数据完整无缺失，当出现夹噪、坏包的情况将不允许

其访问服务端获取数据，这一点同时也用以保障服务可用性，服务器仅接受密钥正确的请求，即便有发生偶然性被碰撞成功，被使用同一请求对服务器进行爆破攻击，这种爆破也会因为密钥的定时迁移而无法持续，从而保证了服务的可用性。

本次我们的创新体现在两点：Scanner加密和动态加密。现在致力于保护数据库安全、节点安全这样的数据集中区的信息安全构思有非常的多，然而我们想要在信息在“传递/向服务器申请”的环节上进行加固的保护，即进行Scanner的信息保障处理。通俗一点比喻起来就是货物来源的邮局和货物送达的目标地点都是非常安全的，然而大家往往忽略了在路上可能被抢包的可能性，丢包可以重发，然而如果包被居心叵测的人拿走获取信息，那是非常具有隐患令人担忧的。所以我们在包裹上安装上电子锁，锁的密码在收发双方都有一套密码本，密码会随着时间而变化，在每个变化的时间间隔内，试出密码的可能性微乎其微，即便可能被试出密码，在下次更替的时限内，也不足以对服务器造成影响，从而达到信息安全与保护的目的。

关于应用方面，使用跨平台的Java语言，可以灵活地应用于各种生产、生活情境中，更有助于为Scanner制作的保障体系通过烧录进Linux或Android为OS的单片机中实现移动终端设备的保障。在各种工作场合下，它都可以通过不同的情景模式调节，发挥更加贴合情境的作用。可以通过当前情境下，三种传输过程的安全级别，抑或是速度、数量的需求决定当前所需的情境。

# 第一章 作品概述

依据物联网的定义，物联网是一种虚拟网络与现实世界实时进行信息交互的新型系统。它的核心和基础仍然是互联网，是在互联网基础上的延伸和扩展。物联网的特点是无处不在的数据感知、以无线为主的信息传输以及智能化的信息处理。由于物联网的上述特点，在信息安全的方面，也不可避免地出现了一些新的特点，这便是本作品构想出现的背景：（以下的 4 点归纳引用及参考的论文出处附于文末）

## 1. 设备、节点等无人看管，容易受到物理操纵

物联网常常被用以代替人完成一些复杂、危险和机械的工作。正因如此，物联网中的设备、节点的工作环境大都无人监控。因此，攻击者就可以轻易接触到这些设备，从而对设备或者嵌入其中的传感器节点进行破坏。攻击者甚至还可以通过更换设备的软硬件，对它们进行非法操控。例如，远程输电过程中，电力企业就可以使用物联网来远程操控一些变电设备。由于缺乏看管，攻击者可轻易地使用非法装置来干扰这些设备上的传感器。可想而知，假如变电设备的某些重要参数被篡改，后果将不堪设想。

## 2. 信息传输主要靠无线通信方式，信号容易被窃取和干扰

物联网在信息传输中多使用无线传输方式，暴露在外的无线信号很容易成为攻击者窃取和干扰的对象，会对物联网的信息安全产生严重的影响。例如：现在的二代身份证都嵌入了 RFID 标签。在使用过程中攻击者可以通过窃取感知节点发射的信号，来获取所需要的信息，甚至是用户的机密信息，假如据此来伪造身份，其后果不堪设想。同时，攻击者也可以在物联网无线信号覆盖的区域内，通过发射无线电信号来进行干扰，从而使无线通信网络不能正常工作甚至瘫痪。比如在物流运输过程中，嵌入在物品中的标签或读写设备的信号受到恶意干扰，就会很容易造成一些物品的丢失。

## 3. 出于低成本的考虑，传感器节点通常是资源受限的

物联网的应用中通常需要部署大量的传感器，以实现充分覆盖特定区域。而且，对于已经部署了的传感器，通常是不会进行回收或维护的。因为量大和一次性的特点，传感器必须具有较低的成本，这样大规模使用才是可行的。为了降低成本，传感器通常是资源受限的。传感器一般体积较小，其能量、处理能力、存储空间、传输距离、无线电频率和带宽也是受限的。例如，MICA 2MPR 400CB 是目前主流的传感器，它的

CPU 主频为 4M 赫兹，内存为 128KB，频率为 916 赫兹，每秒仅仅能传输几十 KB 的数据，传输距离大约为 500 英尺。由于上述种种原因的限制，传感器节点无法使用较复杂的安全协议，因而这些传感器节点或设备也就无法拥有较强的安全保护能力。因此，攻击者针对传感器节点的这一弱点，可以通过采用连续通信的方式使节点的资源耗尽。

#### 4. 物联网中物品的信息能够被自动地获取和传送

物品的感知是物联网应用的前提。物品与互联网相连接，通过 RFID（射频识别）、传感器、二维识别码和 GPS 定位等技术能够随时随地且自动地获取物品的信息。这样，人们随时随地都可以获取物品的位置及周围环境等信息。然而在物联网的应用中，RFID 标签能被嵌入任何物品中。一旦被嵌入人们的日常生活用品中，如果物品的使用者没能察觉，那么物品的使用者将会不受控制地被扫描、定位及追踪。这无疑对个人的隐私构成了极大的威胁。

在此背景下，我们所提出的作品构思如下：

概括而言之，即对“从原始信息到有效数据的转换”过程的保护，并初步形成一个具有预设值的自定义信息安全体系。基于动态口令这种定时更新的动态密码防护，相当于对试图盗取信息包破译的 hacker 规定了极其严苛甚至不可能完成的时间限。在信息传递过程中正常状态下所经历的最长延时适当放宽一个容忍值，以这个时间长度作为间隔，进行动态口令更新，对破译增加了极大的难度。

以此作为核心，在<Sensor-Scanner>（预设AES），<Scanner-Server>（预设MD5-Hash），<Server-Scanner>（预设AES）三个传输过程中，根据情境的不同（所需的安全等级/处理效率的要求不同），用户可自定义加密的方式以应对不同的需求（动态口令中所使用的MD5并非国有的Hash算法，为了更进一步的安全性，后续还将研究使用国有的Hash算法的SM3算法来进行替换）。

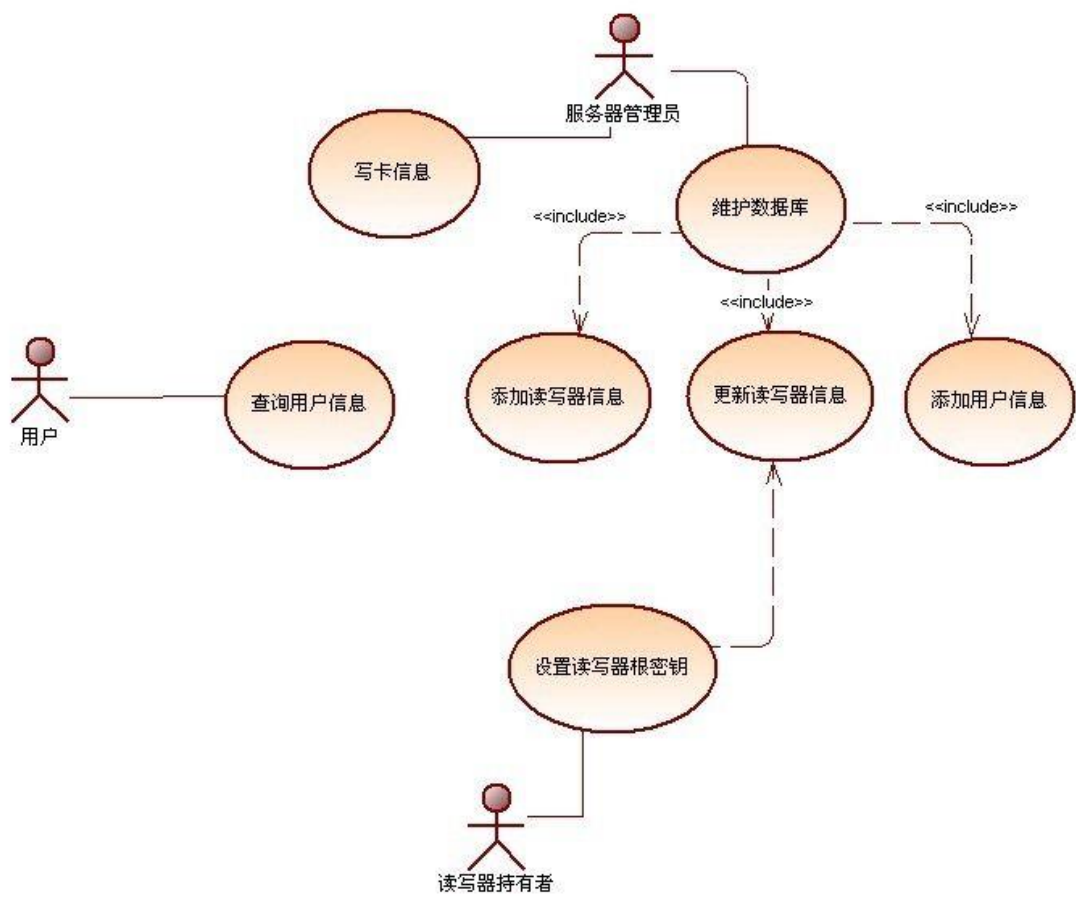
而在应用前景，由于使用的是跨平台语言中的Java语言，并且JavaEE+Server+SQL的组合向来被广泛应用，故可以灵活地应用于各种生产、生活情境中。使用Java的另一个好处在于，我们为Scanner制作的保障体系可以通过烧录进Linux或Android为OS的单片机中实现移动终端设备的保障。不仅仅局限于易于实现、便于移植的方面，在

各种工作场合下它都将通过不同的情景模式调节，发挥更加贴合情境的作用：如需要快速大量验收入库信息的仓储系统，由于入库必然是在当前Scanner与同一数据库（当前纳入的库）间的重复询问，且两者在物理上距离非常相近，安全性较高，便可以直接为Scanner设置（设置需要与服务器相同的key）为与服务器间的验证为“单次认证-多次重复操作”，可以在保障安全性的同时达到高速的目的；另一个追求高安全性的方面比如购物付款的情境下，和人们的钱包相关的东西自然需要极高的安全度，此时三个传输过程均设置为最高级别的保障协议，由于三个过程具有串行性，且两两之间并不相同，无论其中有任何一个没有被攻破，都无法到达最终的Hack目的，更何况，还有着并不知道具体刷新时间的动态时间限的阻拦，并不太高的正向计算的计算复杂度在时间上完全不会令人等候，如上便完成了一次高安全保障、高运行速度的付款行为；除此之外还有很多，我们完全可以通过当前情境下，三种传输过程的安全级别，抑或是速度、数量的需求来决定当前所需切换的情境模式，而且，这一切操作都并不困难，只需轻轻一点，在掌中的终端之上。

## 第二章 作品设计与实现

关于作品的设计与实现，为了有助于更清晰地理解我们的思路，此处将以由浅至深的顺序，按照“用例图-程序流程图-类图”的顺序来依次说明。

首先为用例分析，用例以三个不同身份来依次说明在体系中他们可以执行的操作，并分析各个操作对系统造成的影响，用例图中主要为介绍功能操作，暂不对中/底的层信息安全保障部分进行讨论，用例图如下：



查询用户数据：

放卡至读写器上，在所有的操作中选择“查询用户信息”功能，即在屏幕上显示用户信息，前提是用户信息存在于数据库中并且使用的是服务器授权的读写器。

设置读写器根密钥：



服务器授权的读写器在出厂时就有不可更改的读写器序列号和可更改的初始根密钥，这两个信息在数据库中都已经存在，当然根密钥是可以更改的，在联网的状态下输入新的根密钥，服务器会更新相应的读写器信息。

写卡信息：

往卡内写入用户的标示符（这里是学生学号），这里的标示符是用AES加密过的，加密密钥不可更改，并存在于读写器中，读写器可以对加密后的标示符进行解密。标示符可以用用户在数据库中查找对应用户的信息

维护数据库：

添加读写器信息：读写器出厂前，将读写器的序列号和初始根密钥写入数据库中，初始根密钥需要AES加密后存入数据库中

更新读写器信息：如果读写器持有者对读写器的根密钥做了改动，则数据库也要做相应的改动

添加用户信息：将用户的信息加入到数据库中，包括用户的学号（标示符），名字，性别和现有金额，除了性别以外，所有的信息都将用AES加密

用例图之后，我们将通过程序流程图，来分析用例中的各个操作背后，程序内部执行的流程及数据流向，将表象操作扩展为前端后台进行的一切传递与计算。

如下图中所示，清晰地将程序从开始执行，至循环节开始循环，不同的选择或分支会引导程序的执行走向等体现出来，其中以粗体标明的为前端中可由使用者看见的表象显示。为了令各个模块间互不干扰、各司其职，且不同模块间互相交互时可以更好的契合，仿照网络协议的分层理念，将程序分为三个层，分别为UI层（用户操作界面/交互层）、Scanner层（信息扫描传译器/链接层）以及Server层（服务端数据汇总/数据层）。需要说明的是，操作1-1是服务器端的数据管理与更新，是具有服务器管理员权限的操作，操作1-2是在服务端添加当前Scanner的备案信息（新增Scanner，即一个新的用户为自己开设一个自己拥有密钥及权限的Scanner），操作2-x（为情景模拟项，为方便进行用例测试，x取值暂为1至3）。此处散列表Hash为动态口令Hash，传参为根密钥生成的子密钥，卡的ID号及当前时间。

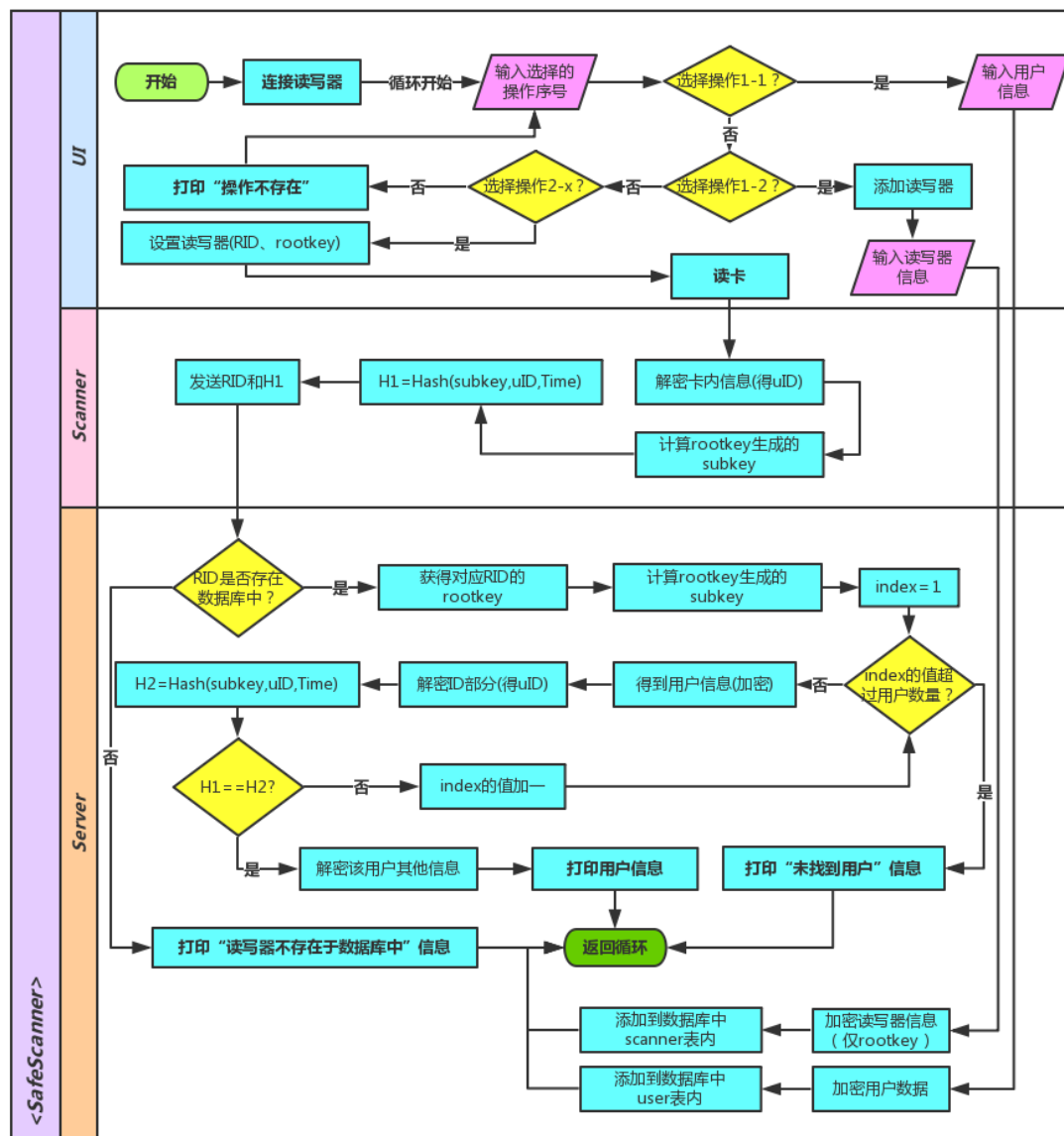
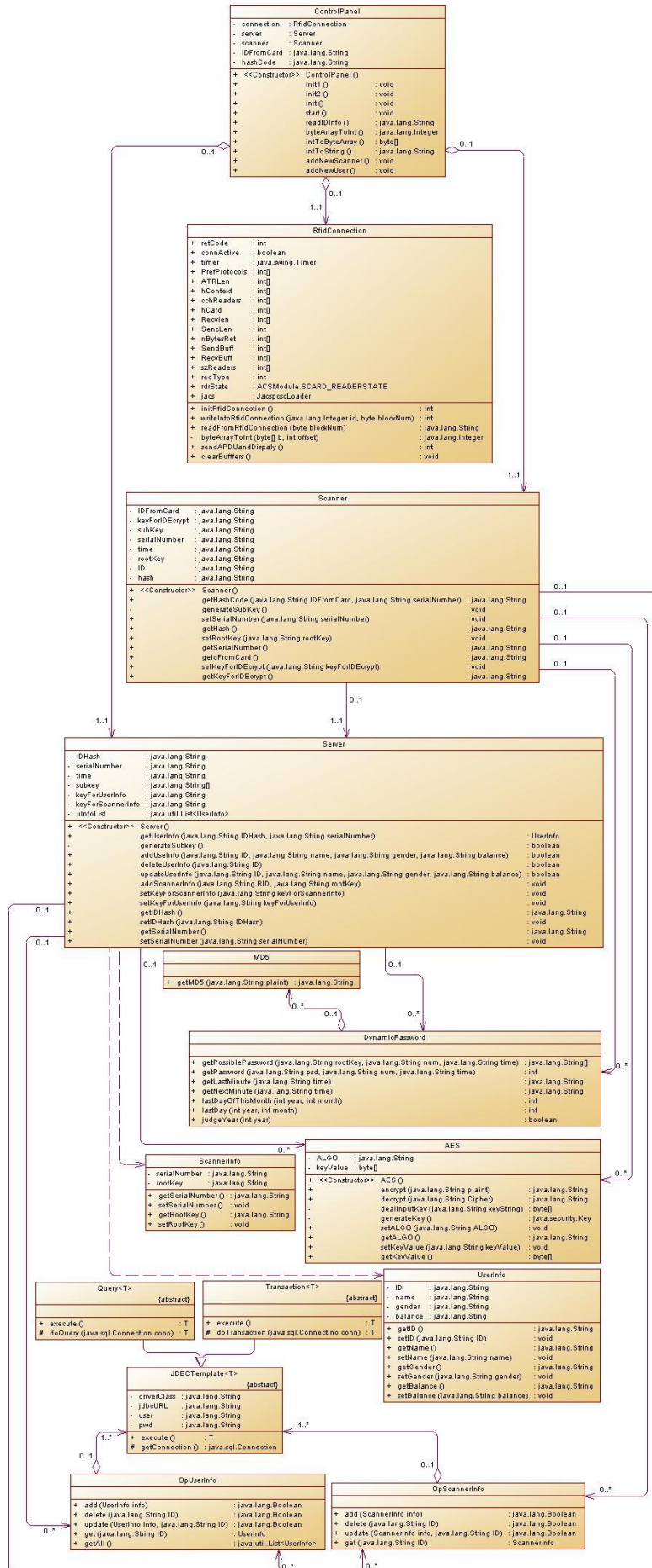


Figure 1.1 SS(SafeScanner)程序流程图

上图中，“ $H1==H2?$  Decode():index++”的分支意为遍历，实际上使用二分查找会极大程度提升时间复杂度，使用遍历的原因是为了防止被攻击者脱库之后通过遍历轻易获得全部有效数据，服务端数据库除了常用的AES加密之外，依照ID进行乱信息填充，有效信息进入时为复写，可以大大提高服务器内信息安全度。

紧接着是如上流程图的具体实现，以类图的方式详细解析实现的方式及各个实现类之间的关系，类图如下所示，对类图的详细解释见后续说明：



类图解释：（自顶向下）

- ControlPanel:

模拟传输信道和实际场景，类里只有唯一的Server、Scanner和RFIDConnection，把Server、Scanner和RFIDConnection功能整合起来，将信息从一个RFIDConnection传输到Scanner，信息经过处理之后又传输到Server端，Server经过验证，确定是否是真实有效的信息，返回相应的信息（“未找到用户”或“非授权读写器”或用户信息）

- RFIDConnection:

用于处理读写器和卡之间的连接关系，有“未放卡”、“写”、“读”、“等待操作”四种状态。

1. “写”：服务端下达写指令，将相应的字符串写进卡的特定区域中
2. “读”：用户下达“获得用户信息”指令，卡内信息会被读进读写器中
3. “未放卡”：读写器已经联网，但是读写器上没有待操作的卡
4. “等待操作”：读写器已经联网，并且有卡放置在其之上，正在等待下达写指令或读卡内信息指令

- Scanner:

读写器内部的操作，包括：

1. 接受到卡内信息，用预设好的解密秘钥解密卡内信息，得到用户ID
2. 得到当前系统时间和此时的根秘钥rootKey生成的子秘钥subKey,把二者和用户ID做MD5操作，得到摘要，将摘要和读写器的序列号一起传送给Server类
3. 设置读写器根秘钥
4. 查看读写器根秘钥
5. 查看读写器序列号

- Server:

Server端执行的操作，包括：

1. 接收到读写器传来的摘要和读写器序列号
2. 通过读写器序列号找到数据库中对应的根秘钥
3. 将当前的根秘钥rootKey生成子秘钥subKey，并得到当前系统时间time、上一分钟的时间timebefore和下一分钟的时间timelater。
4. 将用户数据全部取出，解密用户数据的ID部分，并一一拿来与subKey和time做MD5

操作,得到的摘要与读写器传来的摘要做比较,如果相等则解密用户信息并打印信息,如果不相等,则先用上一分钟和下一分钟的时间分别再做MD5操作,再分别比较,如果有相等,返回用户信息,如果还是不匹配,则比较下一个用户

5. 添加用户信息到数据库
6. 添加读写器信息到数据库
7. 更新读写器信息到数据库

- DynamicPassword:

生成动态口令,核心为MD5算法,输入为序列号、密钥和时间,通过当前的分钟数,利用设定好的函数选取3个字节,转成十进制之后模1000000得到一个不超过6位的数字,即为我们得到的动态口令,并返回这个动态口令(转成String类型后,不足六位的动态口令会在左边用0补齐)

包括的public操作有:

1. 生成服务端当前时间对应的动态口令
2. 生成包括上下误差一分钟所对应的动态口令数组(当前分钟,下一分钟,上一分钟)

- MD5:

输入字符串,生成字符串所对应的MD5摘要

- AES

加密字符串,解密用AES加密的密文

包括的public操作有:

1. 加密输入的字符串
2. 解密输入的字符串
3. 设置加密算法(即可以更改为DES等加密方式)
4. 设置用于加密解密的密钥

- ScannerInfo:

Javabean,以Scanner信息为对象所设定的一个类,信息有Scanner的序列号和所对应的根密钥,操作为两个信息的getter和setter操作

- UserInfo:

Javabean,以用户信息为对象所设定的类,包括用户ID、用户姓名、性别、对应金额

以及它们的getter和setter操作

- OpScannerInfo:

用于操作数据库中的scanner表的信息，包括增删改查四个操作

- OpUserInfo

用于操作数据库中的user表的信息，包括增删改查四个操作

- JDBCTemplate

抽象类，数据库操作模板，

用于连接Mysql数据库

定义了一个抽象方法execute(), execute()是所有数据库操作的抽象方法

- Transaction (事物)

抽象类

1. 实现了execute()方法为正常时提交事物，提交时遇到错误时可以回滚

2. 定义了抽象方法doTrancation(Connection conn),用于事务的处理

- Query(查询)

抽象类

1. 实现了execute()方法，提交用户的查询语句

2. 定义了抽象方法doQuery(Connection conn),用于查询语句的处理

## 第三章 作品测试与分析

为正常运行，推荐使用环境如下

硬件环境：

- Windows 操作系统的计算机( XP/Win7/Win8/Win8.1 ) 一台
- NFC 读写器( 型号为 ACR122 ) 一台或多台
- mifare IC 卡 一张或多张 (所使用的为 13.56MHZ 高频复旦微 FM11RF08 芯片卡)

软件环境：

- IDE: Eclipse or MyEclipse
- Mysql 任意版本

系统配置：

- JAVA\_HOME jdk-1.6 (此处使用 1.6 的老版本是为了配合所使用的 NFC-SDK)
- /Windows/System32 文件夹中更新上传 Jacspcsc.dll 文件

实际测试环境 (Debug 过程中使用环境 及 如下测试方案运行环境)

- Win8.1 Professional (Dell inspiron m421R)

接下来展示测试方案，包含多条测试用例 (包括过程及输出) 及相关解释说明：

(注明:此段测试方案有屏幕录制与视频解说两段视频，见“其他”文件夹内附件)

测试用例1：

1. 将读写器连接上网 (这里是通过接口连接至PC端)，但不放置卡1
2. 将卡放置在读卡器上，测试连接效果

```
Java EE - SafeRFID2/src/Main.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Markers Properties Servers Data Source Explorer Snippets Console
Establish context and obtain hContext handle — status: 0
List PC/SC card readers installed in the system — status: 0
find scanner : ACS ACR122 0
connection failed : -2140434967

Establish context and obtain hContext handle — status: 0
List PC/SC card readers installed in the system — status: 0
find scanner : ACS ACR122 0
connection failed : -2140434967

Establish context and obtain hContext handle — status: 0
List PC/SC card readers installed in the system — status: 0
find scanner : ACS ACR122 0
connection failed : -2140434967

Establish context and obtain hContext handle — status: 0
List PC/SC card readers installed in the system — status: 0
find scanner : ACS ACR122 0
connection failed : -2140434967

Establish context and obtain hContext handle — status: 0
List PC/SC card readers installed in the system — status: 0
find scanner : ACS ACR122 0
connection failed : -2140434967

Establish context and obtain hContext handle — status: 0
List PC/SC card readers installed in the system — status: 0
find scanner : ACS ACR122 0
connection failed : -2140434967

Establish context and obtain hContext handle — status: 0
List PC/SC card readers installed in the system — status: 0
find scanner : ACS ACR122 0
connection failed : -2140434967

Successful connection to: ACS ACR122 0
```

```
Java EE - SafeRFID2/src/Main.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Markers Properties Servers Data Source Explorer Snippets Console
Main [Java Application] C:\Program Files (x86)\Java\jdk1.6.0_10\bin\javaw.exe (2015年6月2日 下午8:25:00)
find scanner : ACS ACR122 0
connection failed : -2140434967

Establish context and obtain hContext handle — status: 0
List PC/SC card readers installed in the system — status: 0
find scanner : ACS ACR122 0
connection failed : -2140434967

Establish context and obtain hContext handle — status: 0
List PC/SC card readers installed in the system — status: 0
find scanner : ACS ACR122 0
connection failed : -2140434967

Establish context and obtain hContext handle — status: 0
List PC/SC card readers installed in the system — status: 0
find scanner : ACS ACR122 0
connection failed : -2140434967

Establish context and obtain hContext handle — status: 0
List PC/SC card readers installed in the system — status: 0
find scanner : ACS ACR122 0
Successful connection to: ACS ACR122 0

后台操作:
(1-1) 添加用户信息
(1-2) 添加读写器信息

情景模拟:
(2-1) 使用默认读写器1(服务器中存在)
(2-2) 使用默认读写器2(服务器中不存在)
(2-3) 自定义读写器

输入 q 退出程序
输入您想要的操作的序号:
>>
```

结果:

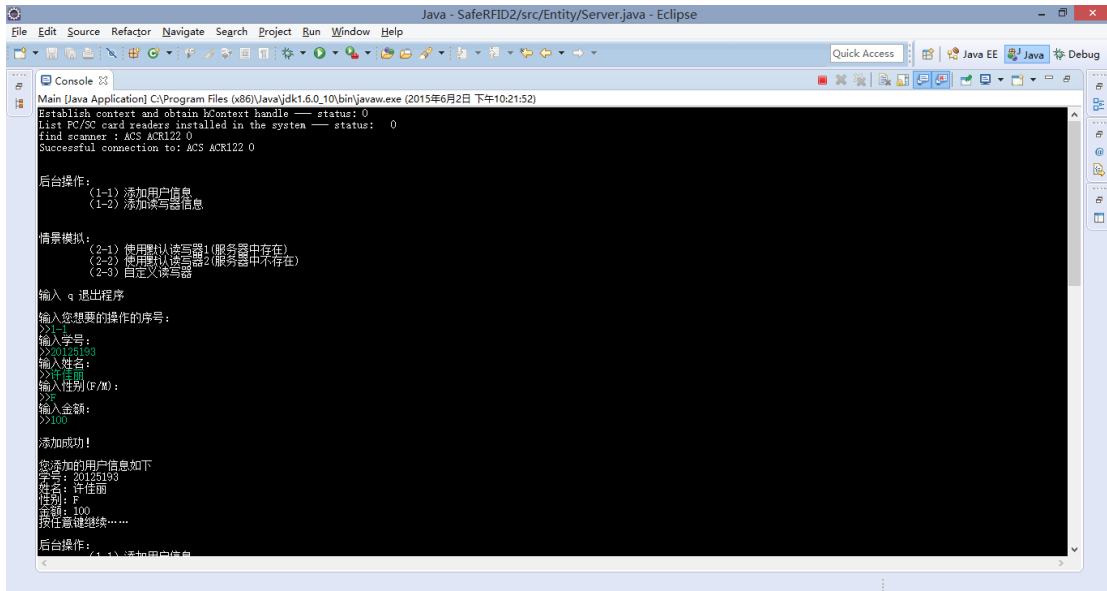
1. 每隔一定时间重复连接卡失败信息（每隔一定的时间会重新尝试读卡连接，并打印读卡连接的结果）
2. 失败信息包括读卡器连接状态字（status），连接上的读卡器型号，以及卡连接失败字样
3. 卡放置在读卡器上后，打印出的信息的最后一行字变成了“Successful Connection”，表示连接成功
4. 连接成功后，显示可供选择的操作提示，并显示“>>”输入提示符，表示按要求输入需要的字符串。



测试用例2:

输入用户信息，并将用户信息添加到数据库中

操作：选择在输入提示符“>>”之后写入“1-1”并回车，表示选择“1-1 添加用户信息”操作（此项若在移动终端上可简单地通过VK映射实现选择操作）



```
Java - SafeRFID2/src/Entity/Server.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access Java EE Java Debug
Main [Java Application] C:\Program Files (x86)\Java\jdk1.6.0_10\bin\javaw.exe (2015年6月2日 下午10:21:52)
Establish context and obtain hContext handle --- status: 0
List PC/SC card readers installed in the system --- status: 0
find scanner : ACS ACR122 0
Successful connection to: ACS ACR122 0

后台操作:
(1-1) 添加用户信息
(1-2) 添加读写器信息

情景模拟:
(2-1) 使用默认读写器1(服务器中存在)
(2-2) 使用默认读写器2(服务器中不存在)
(2-3) 自定义读写器

输入 q 退出程序
输入您想要的操作的序号:
>>1
输入学号:
>>20125193
输入姓名:
>>许佳丽
输入性别(F/M):
>>F
输入金额:
>>100
添加成功!
您添加的用户信息如下
学号: 20125193
姓名: 许佳丽
性别: F
金额: 100
按任意键继续.....
后台操作:
(1-1) 添加用户信息
```

结果:

按顺序输入需要的信息后，显示“添加成功”字样，并打印添加的信息到屏幕上（学号为所输入项的ID信息已经被写入卡的存储空间中，若之后能够成功返回这段用户信息，则证明可通过读卡获得需要的用户信息）

测试用例3:

添加新的读写器信息，包括序列号和根密钥(新增/更改同Owner下的Scanner操作)

操作：在“>>”输入提示符之后输入“1-2”，代表选择“1-2 添加读写器信息”操作

```
Java - SafeRFID2/src/Entity/Server.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Main [Java Application] C:\Program Files (x86)\Java\jdk1.6.0_10\bin\javaw.exe (2015年6月2日 下午10:21:52)
Recv: FF 81 00 19 04
Send: FF 81 00 19 04
Review: 00 51 3D 3D 90 00
Read Block Success!

您的用户信息如下:
ID: 20125193
姓名: 许佳丽
密码: 123456
积分: 100
按任意键继续.....

后台操作:
(1-1) 添加用户信息
(1-2) 添加读写器信息

情景模拟:
(2-1) 使用默认读写器1(服务器中存在)
(2-2) 使用默认读写器2(服务器中不存在)
(2-3) 自定义读写器

输入 q 退出程序
输入您想要的操作的序号:
>>1
输入序列号:
>>123456
输入根密钥:
>>12345678901070504
添加成功!

您添加的读写器信息如下
ID: 1
名称: f1234
根密钥: 12345678901070504
按任意键继续.....
```

结果:

成功添加后，并显示添加的读写器信息（该读写器信息是数据库中唯一存在的信息，为了后面的测试，这个读写器信息也就是2-1操作中默认的读写器信息。在之后的读卡操作中，只能用这个读写器信息才能成功读取卡内信息）

测试用例4:

读卡操作，使用服务端授权的读写器进行读卡（即读写器信息在数据库中是存在的）  
操作：在“>>”输入提示符之后输入“2-1”，代表选择“2-1 用默认读写器1”进行读卡操作

```
Java - SafeRFID2/src/Entity/Server.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Main [Java Application] C:\Program Files (x86)\Java\jdk1.6.0_10\bin\javaw.exe (2015年6月2日 下午10:21:52)
Recv: FF 81 00 19 04
Send: FF 81 00 19 04
Review: 00 51 3D 3D 90 00
Read Block Success!

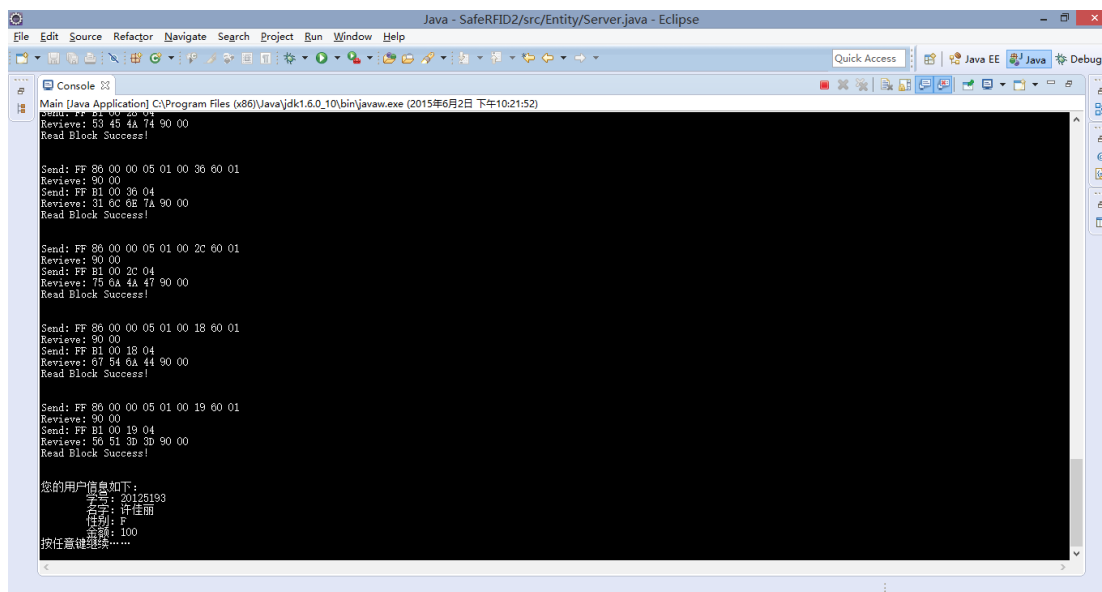
您的用户信息如下:
ID: 20125193
姓名: 许佳丽
密码: 123456
积分: 100
按任意键继续.....

后台操作:
(1-1) 添加用户信息
(1-2) 添加读写器信息

情景模拟:
(2-1) 使用默认读写器1(服务器中存在)
(2-2) 使用默认读写器2(服务器中不存在)
(2-3) 自定义读写器

输入 q 退出程序
输入您想要的操作的序号:
>>1
输入序列号:
>>123456
输入根密钥:
>>12345678901070504
添加成功!

您添加的读写器信息如下
ID: 1
名称: f1234
根密钥: 12345678901070504
按任意键继续.....
```



```
Java - SafeRFID2/src/Entity/Server.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access Java EE Java Debug

Main [Java Application] C:\Program Files (x86)\Java\jdk1.6.0_10\bin\javaw.exe (2015年6月2日 下午10:21:52)
Send: FF 86 00 00 05 01 00 36 60 01
Receive: 90 00
Send: FF 81 00 36 04
Receive: 81 0C 0E 7A 90 00
Read Block Success!

Send: FF 86 00 00 05 01 00 2C 60 01
Receive: 90 00
Send: FF 81 00 2C 04
Receive: 75 0A 4A 47 90 00
Read Block Success!

Send: FF 86 00 00 05 01 00 18 60 01
Receive: 90 00
Send: FF 81 00 18 04
Receive: 87 54 6A 44 90 00
Read Block Success!

Send: FF 86 00 00 05 01 00 19 60 01
Receive: 90 00
Send: FF 81 00 19 04
Receive: 86 51 3D 3D 90 00
Read Block Success!

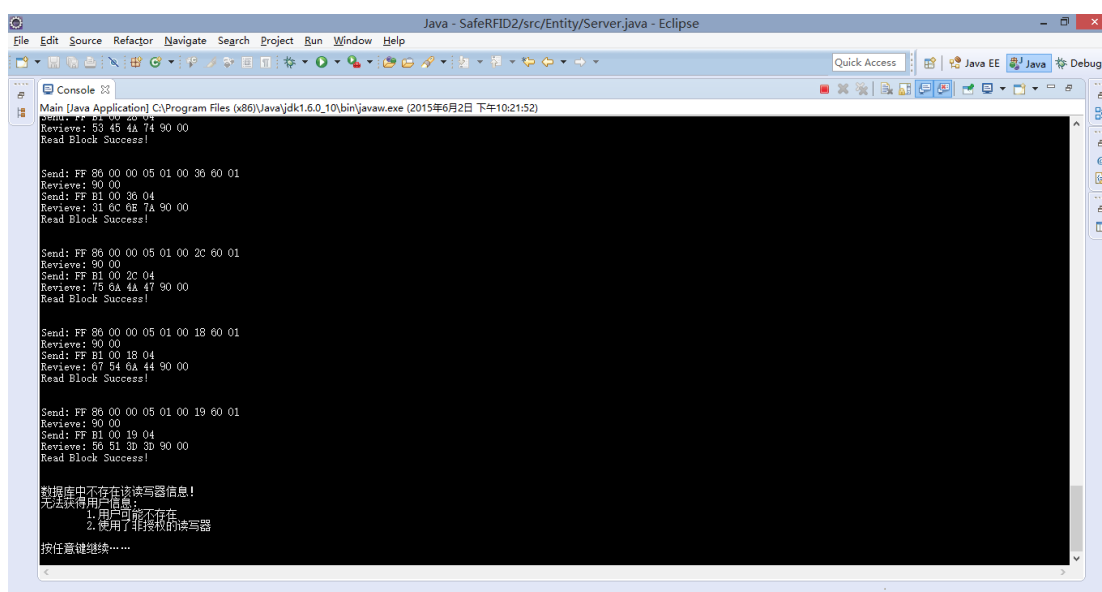
您的用户信息如下:
账号: 20125193
姓名: 许佳丽
性别: F
金额: 100
按任意键继续.....
```

结果:

1. 打印Scanner类中调用“读卡操作”时分别给读写器发送的指令和接收到的指令，因为用户的ID信息分别放在了卡的六个不同的扇区中，故读卡操作进行了六次，每次读不同的扇区。
2. 最后打印出卡内的ID经过一系列处理后，从数据库中得到的用户信息，可以看到，就是我们之前添加的用户信息

测试用例5:

如果使用了未授权的读卡器（即读卡信息在数据库中不存在），观察读卡的结果操作：在“>>”输入提示符之后输入“2-2”，代表选择“2-2 使用默认读写器2”



```
Java - SafeRFID2/src/Entity/Server.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access Java EE Java Debug

Main [Java Application] C:\Program Files (x86)\Java\jdk1.6.0_10\bin\javaw.exe (2015年6月2日 下午10:21:52)
Send: FF 86 00 00 05 01 00 36 60 01
Receive: 90 00
Send: FF 81 00 36 04
Receive: 81 0C 0E 7A 90 00
Read Block Success!

Send: FF 86 00 00 05 01 00 2C 60 01
Receive: 90 00
Send: FF 81 00 2C 04
Receive: 75 0A 4A 47 90 00
Read Block Success!

Send: FF 86 00 00 05 01 00 18 60 01
Receive: 90 00
Send: FF 81 00 18 04
Receive: 87 54 6A 44 90 00
Read Block Success!

Send: FF 86 00 00 05 01 00 19 60 01
Receive: 90 00
Send: FF 81 00 19 04
Receive: 86 51 3D 3D 90 00
Read Block Success!

数据库中不存在该读写器信息!
无法获得用户信息:
1. 用户ID不存在
2. 使用了非授权的读写器
按任意键继续.....
```

结果:

1. 虽然读卡成功,但是得不到用户的信息(由于Scanner防护,服务器并不会受理)
2. 打印错误的最有可能的原因,和其他可供参考的原因

综上所述,测试用例均以最初规划中一样正常运行,Scanner对Sensor(NFC)中信息阅读准确,对Server衔接紧密,过程中加密/解密算法高效执行,非合法询问在Scanner端便被阻止,不会对服务器造成影响。

测试用例表明,项目可遵循体系稳定正确地运行。

## 第四章 创新性说明

Scanner加密和动态加密是本作品两个核心的创新点所在。

现在致力于保护数据库安全、节点安全这样的数据集中区的信息安全构思有非常的多，然而我们想要在信息在“传递/向服务器申请”的环节上进行加固的保护，即进行Scanner的信息保障处理。通俗一点比喻起来就是货物来源的邮局和货物送达的目标地点都是非常安全的，然而大家往往忽略了在路上可能被抢包的可能性，丢包可以重发，然而如果包被居心叵测的人拿走获取信息，那是非常具有隐患令人担忧的。所以我们在包裹上安装上电子锁，锁的密码在收发双方都有一套密码本，密码会随时间而变化，在每个变化的时间间隔内，试出密码的可能性微乎其微，即便可能被试出密码，在下次更替的时限内，也不足以对服务器造成影响，从而达到信息安全与保护的目，同时为Scanner单调的Reader功能增加了智能化的处理，增强了安全保障。

动态口令（Dynamic Password）的定义为：根据专门的算法生成一个不可预测的随机数字组合，每个密码只能使用一次，目前被广泛运用在网银、网游、电信运营商、电子商务、企业等应用领域。基于令牌和服务器的时间同步，通过运算来生成一致的动态口令，基于时间同步的令牌，但由于其同步的基础是国际标准时间，则要求其服务器能够十分精确的保持正确的时钟，同时对其令牌的晶振频率有严格的要求，从而降低系统失去同步的几率，从另一方面，基于时间同步的令牌在每次进行认证时，服务器端将会检测令牌的时钟偏移量，相应不断的微调自己的时间记录，从而保证了令牌和服务器的同步，确保日常的使用，但由于令牌的工作环境不同，在磁场，高温，高压，震荡，浸水等情况下易发生时钟脉冲的不确定偏移和损坏。在这一点上，我们想出的一个创新点为“正负偏移1”的方式，即服务器核实时，以当前时间戳前后各偏移一位，将此三位与发送至服务器的H码进行比对，由于此处我们使用的动态口令为ASC-6bits，即共计有127的6次方中排列，命中率乘以三依然可以看作是常数级别的变化，对安全性不会带来影响，然而可以实际地解决时间差量的问题。

## 第五章 总结

“物联网”的理念源于比尔盖茨1995 年《未来之路》一书。1999 年，美国Auto-ID 首先提出“物联网”的概念，即把所有物品通过射频识别等信息传感设备与互联网连接起来，实现智能化识别和管理。2005 年11 月17 日，在突尼斯举行的信息社会世界峰会上，国际电信联盟(ITU)发布了《ITU 互联网报告2005：物联网》，正式提出了“物联网”的概念。顾名思义，物联网是“物-物相联的网络”，是指通过射频识别(RFID)、红外传感器、全球定位系统、激光扫描器等信息传感设备，依据约定的协议，把任何嵌入包含其信息的可识别智能芯片的物品与互联网连接起来，进行信息的交换和通信，以实现智能化的识别、定位、跟踪、监控和管理的一种网络。

物联网在网络上被评价为是下一个“万亿元”的产业，它得到了各国政府的高度重视。有报道称美国政府甚至将其作为重振经济的法宝，我国也将物联网列入了新兴战略产业。然而我们在乐观看待物联网发展的同时，在其应用中可能出现的信息安全问题也绝不容忽视。我们应该关注物联网信息安全，它在成长的过程中亟待呵护，一步一个脚印踏实地前行，考虑到每一种可能带来隐患的可能性，争取将我们自己的物联网建设成稳定、令人信赖的新生活方式。

这次我们为物联网的最前端——传感网的防护，提出了一点可能并不足够成熟的想法，然而我们实际去操作、去研究、去尝试了，这是一个我们想到的传感网络的安全体系，这是一个我们做出的实现原型(或者说是一个DEMO)，它连接了前端的Sensor、后端的Server，着重着眼于保护那将两者连接起来的中间的Scanner，这一存在向来仅仅作为简单的硬件前端，有时甚至不区分于Reader，然而Scanner是可以作为移动终端存在的，它可以与单片机或者芯片一起，形成一个具有安全保障的便捷工具，为人们在物联网世界的交互提供方便的服务，自然，更重要的——还有安全的保障。

## 参考文献

- [1] 胡向东. 物联网研究与发展综述 [J]. 数字通信, 2010(02): 17- 21.
- [2] 臧劲松. 物联网安全性能分析 [J]. 计算机安全, 2010(06): 51- 55.
- [3] 郭莉, 严波, 沈延. 物联网安全系统架构研究 [J]. 信息安全与通信保密, 2010(12): 73- 75.
- [4] 吴同. 浅析物联网的安全问题 [J]. 网络安全技术与应用, 2010(08): 7- 8.
- [5] 欧阳常青, 黄生叶, 刘完芳. 低成本RFID安全问题综述 [J]. 网络安全技术与应用, 2007(10): 45- 46.