

《物联网安全》

课程设计文档

姓 名： 刘会禹 学号： 20161000780

姓 名： 贾晨阳 学号： 20161003753

姓 名： 邹鑫娣 学号： 20161000780

姓 名： 罗一飞 学号： 20161004068

班 号：\_\_\_193162\_\_\_ 组长： 刘会禹

院（系）： 计算机学院 专业： 网络工程

指导教师： 姚 宏 职称： 教授

2019 年 4 月

目录

[1.需求分析 1](#_Toc6693952)

[1.1系统概述 1](#_Toc6693953)

[1.2编译环境概述 1](#_Toc6693954)

[1.2Keberos认证 1](#_Toc6693955)

[1.2.1 Keberos认证介绍 1](#_Toc6693956)

[1.2.2具体流程分析 1](#_Toc6693957)

[1. 概要设计 2](#_Toc6693958)

[2.1系统概要分析 2](#_Toc6693959)

[2.1.1物理层面分析 2](#_Toc6693960)

[2.1.2逻辑层面分析 3](#_Toc6693961)

[2.1.3时序图分析 4](#_Toc6693962)

[2.1.4状态分析 4](#_Toc6693963)

[2.2数据库设计 7](#_Toc6693964)

[2.2.1Server端数据库 7](#_Toc6693965)

[2.3.2 AS端数据库分析 9](#_Toc6693966)

[2.3.3 TGS端数据库分析 9](#_Toc6693967)

[2. 具体设计 10](#_Toc6693968)

[3.1数据包封装 10](#_Toc6693969)

[3.1.1数据包封装统一标准 10](#_Toc6693970)

[3.1.2 C→AS IDc|| IDtgs|| TS1 10](#_Toc6693971)

[3.1.3 AS → C Ekc[ Kc,tgs || IDtgs || TS2 || Lifetime2 || Ticket tgs ] 10](#_Toc6693972)

[3.1.4 C → TGS IDV || Ticket tgs || Authenticatorc 11](#_Toc6693973)

[3.1.5 TGS → C EKc,tgs [ Kc,v || IDV || TS4 || Ticket V ] 12](#_Toc6693974)

[3.1.6 C → V Ticketv || Authenticatorc 12](#_Toc6693975)

[3.1.7 V → C EKc,v[ TS5 + 1 ] 13](#_Toc6693976)

[3.1.8应用层报头设计 13](#_Toc6693977)

[3.2具体模块设计 14](#_Toc6693978)

[3.1 Client模块 14](#_Toc6693979)

[3.2 AS模块 15](#_Toc6693980)

[3.3 TGS模块 16](#_Toc6693981)

[3.4 Server模块 16](#_Toc6693982)

[3.3 UI设计 18](#_Toc6693983)

[3.3.1 UI设计环境 18](#_Toc6693984)

[3.3.2 UI功能需求 18](#_Toc6693985)

[3.3.3 UI功能设计 18](#_Toc6693986)

[四.小组分工 27](#_Toc6693987)

# 1.需求分析

## 1.1系统概述

我们所做系统为一个基于Kebesros认证的在线阅读系统。

## 1.2编译环境概述

操作系统:Windows

编译语言:C++

数据库:SQL Server

源代码管理工具:GitHub

## 1.2Keberos认证

### 1.2.1 Keberos认证介绍

Kerberos 是一种网络认证协议，其设计目标是通过密钥系统为客户机 / 服务器应用程序提供强大的认证服务。该认证过程的实现不依赖于主机操作系统的认证，无需基于主机地址的信任，不要求网络上所有主机的物理安全，并假定网络上传送的数据包可以被任意地读取、修改和插入数据。在以上情况下， Kerberos 作为一种可信任的第三方认证服务，是通过传统的密码技术（如：共享密钥）执行认证服务

### 1.2.2具体流程分析

1. 客户端向认证服务器发送访问需求，其中包括自己的身份信息，以及想要申请访问的票据授权服务端的名称。
2. 认证服务器接收到信息之后对服务器端的身份进行认证，认证通过之后将会话密钥Session Key和访问票据Ticket返回给客户端。
3. 客户端将自己的身份信息和从认证服务器获得的票据以及要访问的服务器名称发送给票据授权服务器。
4. 票据授权服务器接收到数据包，确定其身份和票据的真实有效性之后，将客户与服务器的会话钥匙以及访问票据TicketV返回给客户端。
5. 客户端将自己身份信息和从票据授权服务其得到的票据发给服务器。
6. 服务器认证完客户身份后返回一个时间给客户以确定并非消息重放，认证完成。

# 概要设计

## 2.1系统概要分析

### 2.1.1物理层面分析

物理层面，需要AS、TGS、应用服务器和客户端，分别部署在小组四人各自电脑上。客户端可以在多台电脑上开启。

用户每次获取服务都需要按顺序经过步骤（1）（2）（3）。



图 1‑1物理视图

### 2.1.2逻辑层面分析

按照逻辑关系，系统可分为前端、认证层、业务层、数据层、数据库和运行环境共六个模块。前端主要负责UI界面的设计。认证层负责此次系统的大部分内容，即报文制定、加解密、状态机、检验与发放认证、提供服务票据等功能，是整个系统除应用外的核心部分。业务层负责制定应用功能，在此次在线阅读应用设计中，主要完成对用户以及图书的管理。数据层负责对整个应用中的数据进行处理，例如报文的拆包与分析、密钥数据库的访问、图书阅读章节的发送、消息的组织装包等。数据库我们选择的是SQL Server，运行环境为Windows（WIN10）。

详细逻辑分层见下图。

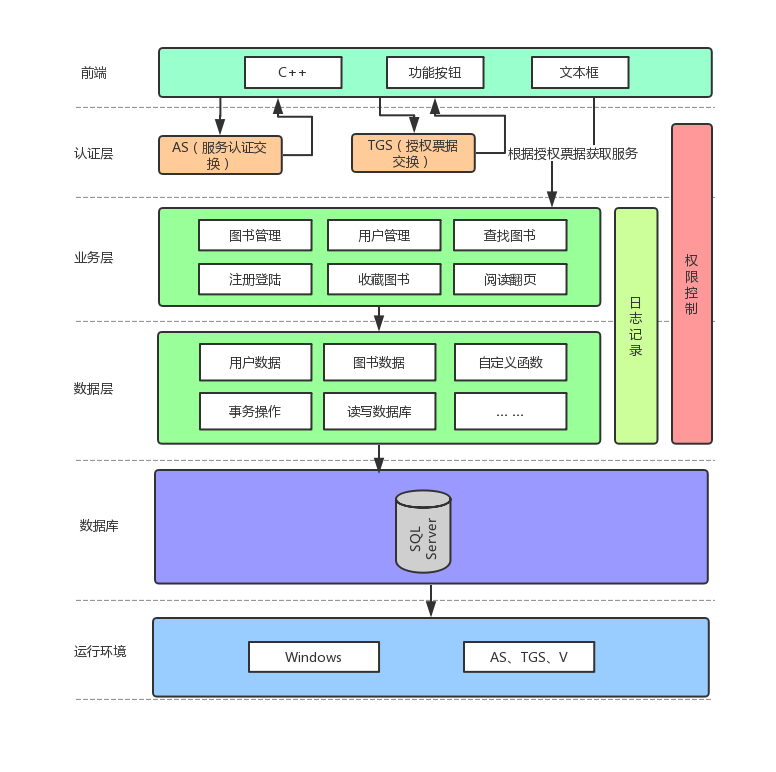


图 1‑2逻辑视图

### 2.1.3时序图分析

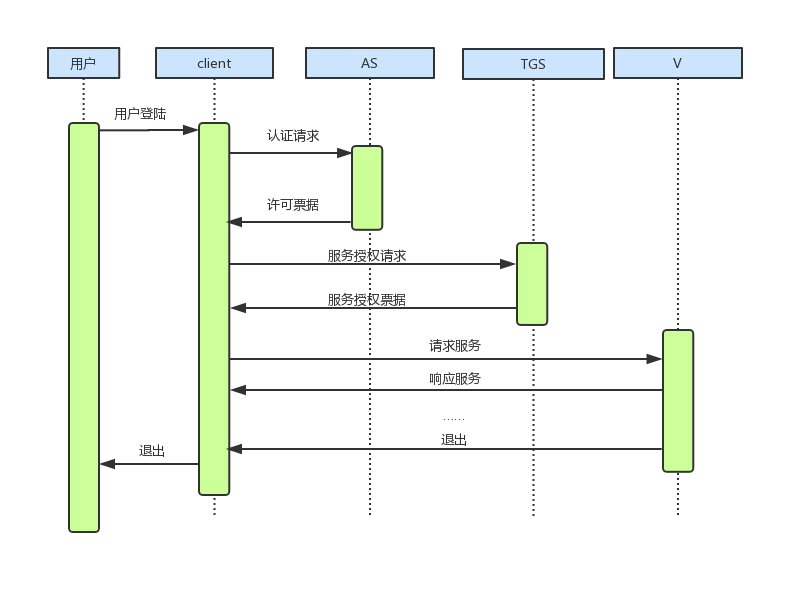


图 1‑3时序图示例

### 2.1.4状态分析

在实际情况中，当然不可能只有一个用户来访问应用，那么针对不同用户在不同时间申请服务的情况，除了多线程外，我们还需要制定状态机来进行任务的确认与分配，表现在这里即是client何时与AS通信、client和TGS通信时是否已经完成了上一步骤等。

根据时序图，我们能够很轻易地得到这样几种状态：

（1）Client—AS 用户与AS交换报文获取许可票据

（2）Client—TGS 在上一步基础下，用户与TGS交换报文获取服务授权

（3）Client—V 在上一步基础下，用户与V交换报文开始访问应用

针对client、AS、TGS、V有：

Client：与AS通信、与TGS通信、与V通信

AS：等待、与client通信

TGS：等待、与client通信

V：等待、与client通信



图 1‑4AS、TGS、V状态

实际上，AS、TGS和V的状态都只有两种，即等待和开启线程开始通信或服务。我们需要做的就是，让client知道自己何时该与他们进行通信。



图 1‑5client状态

如图，对于client，我们可以用两个状态变量DoA、DoT来表示他当前应该做的事情。



图 1‑6状态表

在实践阶段只需要添加DoA、DoT两个变量并进行判断即可。

伪代码示例：

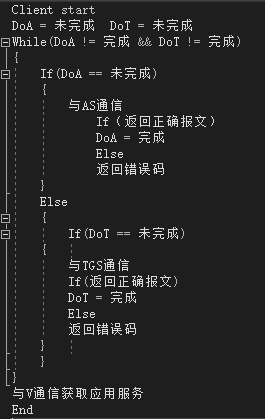


图 1‑7伪代码示例

## 2.2数据库设计

### 2.2.1Server端数据库

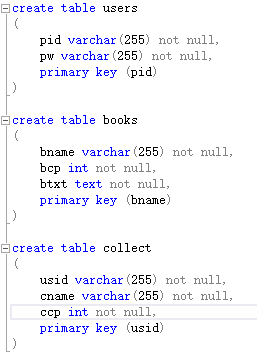


图 1‑8Server端数据库设计

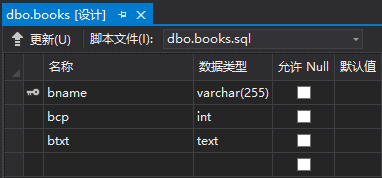


图 1‑9Book表项

bname：书名

bcp：章节页号

btxt：内容

在书籍表中，我们存入一本书的方式是将其根据章节或页号先进行拆分，然后将书名、章节或页号以及该段内容存入一条记录。所有同一bname记录根据章节页码组合在一起才是一本书的所有内容。这样做的好处是，每次请求阅读一本书时，我们可以只传输某章节内容而不用一下子传输完全部，同时使得用户添加书签项更加简单。

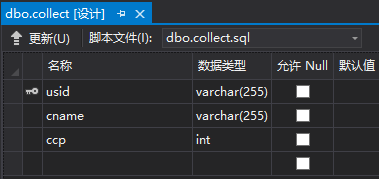


图 1‑10collect表项

usid：用户id

cname：收藏的书籍名称

ccp：收藏书籍书签所在章节或页码

收藏表项配合书籍表，我们能够很容易的做到添加书签功能。对于不同用户，他们的收藏书单都在该表项中，只需根据用户id区分即可。

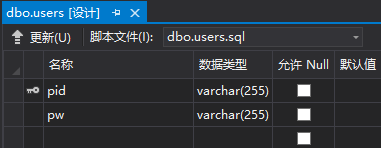


图 1‑11users表项

pid：用户id

pw：密码

暂定用户表项只作为区分收藏功能配合其他表项使用。

### 2.3.2 AS端数据库分析

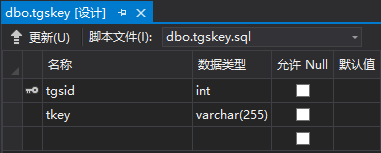


图 1‑12TGSKey表项

tgsid：TGSID

tkey：Client与TGS的会话密钥

### 2.3.3 TGS端数据库分析

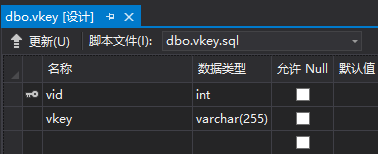


图 1‑13VKey表项

vid：vid

vkey：Client与V的会话密钥

# 具体设计

## 3.1数据包封装

### 3.1.1数据包封装统一标准

以下位数均以char字符位为准,1位char字符即转换8位bit

用户IDc（4位），以0001，0002，0003,0004……依次后延

服务器IDv(4位),以1001,1002,1003,1004……依次后延,设计暂定唯一默认服务器1001

TGSIDtgs(4位),以2001,2002,2003,2004……依次后延,设计暂定唯一默认服务器2001

TS(12位)

Key(8位)

Lifetime(4位)

AD(15位)

### 3.1.2 C→AS IDc|| IDtgs|| TS1

|  |  |
| --- | --- |
| IDc(4位) | IDtgs(4位) |
| TS1(12位) | |

总计32+32+96=160bit

### 3.1.3 AS → C Ekc[ Kc,tgs || IDtgs || TS2 || Lifetime2 || Ticket tgs ]

Ticket tgs格式如下，实际内容经过Key tgs加密

|  |  |  |  |
| --- | --- | --- | --- |
| Kc,tgs(8位) | | IDc(4位) | |
| ADc(15位) | | | |
| IDtgs(4位) | TS2(12位) | | Lifetime2（4位） |
| 0填充（1位） | | | |

总计64+32+120+32+96+32+8(0填充位)=384bit

实发数据部分如下,内容经过Key c加密

|  |  |
| --- | --- |
| Kc,tgs(8位) | IDtgs (4位) |
| TS2(12位) | Lifetime2(4位) |
| Ticket tgs(48位) | |
| 0填充（4位） | |

总计64+32+96+32+384+32(0填充位)=640bit

### 3.1.4 C → TGS IDV || Ticket tgs || Authenticatorc

Autnenticatorc格式如下，实际内容经过Keyc,tgs加密

|  |  |  |
| --- | --- | --- |
| IDc（4位） | ADc（15位） | |
| TS3（12位） | | 0填充（1位） |

总计32+120+96+8(0填充位)=256bit

实发数据部分如下

|  |
| --- |
| IDv （4位） |
| Ticket tgs （48位） |
| Autnenticatorc （32位） |

总计32+384+256=672bit

### 3.1.5 TGS → C EKc,tgs [ Kc,v || IDV || TS4 || Ticket V ]

Ticket V格式如下，实际内容经过Key v加密

|  |  |  |  |
| --- | --- | --- | --- |
| Kc,v(8位) | | IDc (4位) | |
| ADc(15位) | | | |
| IDv(4位) | TS4(12位) | | Lifetime4（4位） |
| 0填充（1位） | | | |

总计64+32+120+32+96+32+8(0填充位)=384bit

实发数据部分如下,内容经过Key c,tgs加密

|  |  |  |
| --- | --- | --- |
| Kc,v(8位) | IDv（4位） | TS4（12位） |
| Ticket V （48位） | | |

总计64+32+96+384=576bit

### 3.1.6 C → V Ticketv || Authenticatorc

Autnenticatorc格式如下，实际内容经过Keyc,v加密

|  |  |  |
| --- | --- | --- |
| IDc（4位） | ADc （15位） | |
| TS5（12位） | | 0填充（1位） |

总计32+120+96+8(0填充位)=256bit

实发数据部分如下

|  |
| --- |
| Ticket v （48位） |
| Autnenticatorc （32位） |

总计384+256=640bit

### 3.1.7 V → C EKc,v[ TS5 + 1 ]

|  |  |
| --- | --- |
| TS5+1（12位） | 0填充（4位） |

总计96+32=128bit

### 3.1.8应用层报头设计

|  |  |  |
| --- | --- | --- |
| 功能码 (4位) | 错误码（4位） | 冗余部分（6位） |

功能码

|  |  |
| --- | --- |
| 0000 | Keberos认证 |
| 0001 | 用户登陆 |
| 0010 | 开始阅读 |
| 0011 | 下一页 |
| 0100 | 上一页 |
| 0101 | 退出 |

错误码

|  |  |
| --- | --- |
| 0000 | 查询数据库失败 |
| 0001 | 登陆密码错误 |
| 0010 | 阅读操作报错 |
| 0011 | 上一页操作报错 |
| 0100 | 下一页操作报错 |
| 0101 | 退出操作报错 |
| 0110 | Lifetime有效期过期 |
| 0111 | 状态繁忙 |
| 1111 | 无误，正常 |

## 3.2具体模块设计

### 3.1 Client模块

**1)public Client(String IDC,string CAddr)**

构造函数，new Client并对成员变量用户ID以及用户IP地址赋值。

**2）public void SokcetLink (string SockAddr,int SockPort);**

套接字通信函数，根据IP端口发起socket连接。

**3）public void Authentication ();**

Keberos认证函数，相继调用AS认证函数，TGS认证函数，应用服务器认证函数，以完成Keberos认证过程，进而保证Client客户端能够与服务器请求服务。

**4) public void LinkAS（string ASAddr,int ASPort）;**

AS交互认证函数，通过套接字通信函数与AS建立socket通信，并通过相应的数据封装

函数和数据解封函数完成与AS的数据验证交互。

**5）public void LinkTGS(string TGSAddr,int TGSPort);**

TGS交互认证函数，通过套接字通信函数与TGS建立socket通信，并通过相应的数据封装函数和数据解封函数完成与TGS的数据验证交互。

**6）public void LinkV（string VAddr,int VPort）;**

应用服务器交互认证函数，通过套接字通信函数与V应用服务器建立socket通信,并通过相应的并通过相应的数据封装函数和数据解封函数完成与客户服务器的数据验证交互。

/\*以下为针对各种数据包的封装，解封装函数\*/

**7) private string C\_ASDataEncapsulation(string IDTGS);**

AS通信数据封装函数，根据Client向AS所需发送的数据进行封装加密。

**8）private string C\_ASDataDeEncapsulation();**

AS通信数据解封装函数，根据AS发来的数据包，进行解密拆分。

**9）private string C\_TGSDataEncapsulation(string IDV,string TicketTGS);**

TGS通信数据封装函数，根据Client向TGS所需发送的数据进行封装加密。

**10）private string C\_TGSDataDeEncapsulation();**

TGS通信数据解封装函数，根据TGS发来的数据包，进行解密拆分。

**11) private string C\_VDataEncapsulation(string TicketV);**

客户服务器通信数据封装函数，根据Client向服务器所需发送的数据进行封装加密。

**12）private string C\_VDataDeEncapsulation();**

客户服务器通信数据解封装函数，根据TGS发来的数据包，进行解密拆分。

/\*以下为实际应用功能函数，实际编写过程可能会有所增添修改\*/

**13)public bool login();**

登陆功能函数。

**14)public bool readbook();**

阅读功能函数。

**15)public bool nextpage();**

下一页功能函数。

**16)public bool prepage();**

上一页功能函数。

**17)public bool collect();**

收藏书籍函数

**18)public void exit();**

退出功能函数。

### 3.2 AS模块

**1)public ASserver(int Port);**

socket等待监听，可实现创建多线程为多client提供服务。

**2）public string GetKeyCTGS(string TGSID);**

查找数据库，根据Client发来的TGSID得到Client与TGS的会话密钥。

**3）private string GetTicketTGS(string KeyCTGS,string IDC,string CAddr,string IDTGS,int LifeTime2);**

封装加密生成Client与TGS验证所需要的TicketTGS。

**4）private string AS\_CDataEncapsulation(string TicketTGS,string KeyCTGS,string IDTGS,int LifeTime2);**

封装加密整合生成最终要发回Client的数据包。

**5）private string AS\_CDataDeEncapsulation();**

将Client发来的数据包进行解封装。

### 3.3 TGS模块

**1）public TGSserver(int Port);**

socket等待监听，可实现创建多线程为多client提供服务。

**2）public string GetKeyCV(string TGSID);**

查找数据库，根据Client发来的VID得到Client与V的会话密钥。

**3）private string GetTicketV(string KeyCV,string IDC,string CAddr,string IDV,int LifeTime4);**

封装加密生成Client与V验证所需要的TicketV。

**4）private string TGS\_CDataEncapsulation(string TicketV,string KeyCV,string IDV,int LifeTime4);**

封装加密整合生成最终要发回Client的数据包。

**5）private string TGS\_CDataDeEncapsulation();**

将Client发来的数据包进行解封装。

### 3.4 Server模块

**1）public Vserver(int Port);**

socket等待监听，可实现创建多线程为多client提供服务。

**2）private string V\_CDataEncapsulation();**

对最终要发回Client的数据包进行封装加密。

**3）private string V\_CDataDeEncapsulation();**

将Client发来的数据包进行解封装。

**4）private void Receive();**

消息响应函数，针对接收到的数据，选择相应的函数进行回应。

/\*以下为应用功能函数，实际编写过程可能会有所增添修改\*/

**5）public bool login(string CID,string CPWD);**

登陆功能函数

**6）public bool readbook();**

阅读功能函数。

**7）public bool nextpage();**

下一页功能函数。

**8）public bool prepage();**

上一页功能函数。

**9）public bool collect();**

收藏书籍函数

**10）public void exit();**

退出功能函数。

## 3.3 UI设计

### 3.3.1 UI设计环境

Microsoft Visual Studio 2017

Visual C++ MFC组件

### 3.3.2 UI功能需求

根据前端需求，UI设计划分为四个部分：

1. 登陆界面
2. 注册界面
3. 书籍预览与收藏界面（主界面）
4. 书籍阅读界面

四个部分实现的功能分别为：

验证登陆、跳转至注册、跳转至主页面、登出功能——登陆界面功能；

注册账户、跳转至登陆界面——注册界面功能；

浏览书籍目录、选择书籍跳转至书籍阅读界面、浏览收藏目录、选择收藏书籍跳转至收藏书籍页面、退出——书籍预览与收藏界面（主界面）功能；

浏览书籍正文、书籍翻页、返回书籍预览与收藏界面（主界面）——书籍阅读界面功能；

### 3.3.3 UI功能设计

**一、登陆界面UI功能设计**

建立MFC项目reading并在该项目下新建MFC类Login，在资源视图的Dialog中选择当前对话框并进行控件布置。所需部件包括：三个Static Text静态文本框分别显示文本“登陆界面”、“账号”、“密码” ； 两个Edit Control编辑框放置文本“账号”、“密码”后作为用户登陆输入区间 ； 三个Button按钮分别是“登陆”、“登出”以及“注册”。

完成控件布置后，为三个Button添加变量

“登陆”——button1

“登出”——button2

“注册”——button3

再为各个button添加响应事件OnBnClicked（）

各自伪代码如下所示

**登陆**：

{

连接数据库table users

数据库查询

if（ edit control（账号）==pid）

{

if（edit control（密码）==pw）

XXX.DoModal(); //跳转至XXX（主页面）对话框

将账号id赋给全局变量logid;

if（edit control（密码）！=pw）

显示“密码错误”通知框

}

else

显示“账号不存在”通知框

}

**登出**

{

OnCancel();

}

**注册**

{

XXX.DoModal(); //跳转至XXX（注册）对话框

}

**登陆界面示意图如下所示：**

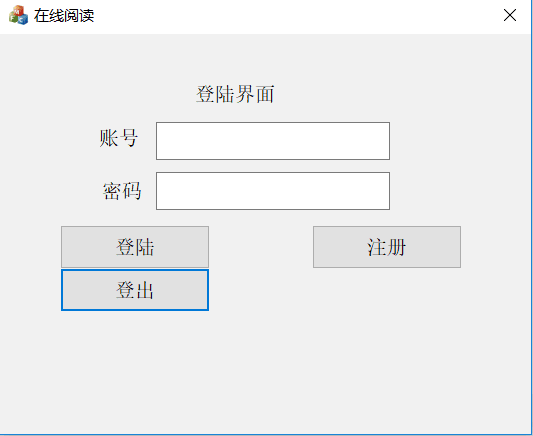


图 3- 1 登陆界面

**二、注册界面UI功能设计**

建立方式同登陆界面，控件布置包括：四个Static Text静态文本框分别显示文本“注册界面、“输入账号”、“输入密码”、“确定密码” ； 三个Edit Control编辑框放置文本“输入账号”、“输入密码”、“确定密码”后作为用户登陆输入区间 ； 两个Button按钮分别是“确定”以及“返回”。

完成控件布置后，为两个Button添加变量

“确定”——button4

“返回”——button5

再为各个button添加响应事件OnBnClicked（）

各自伪代码如下所示

**确定**

{

连接数据库；

数据库查询;

if( edit control（输入账号）存在)

显示“该账号已存在”通知框;

if（ edit control（输入密码）！= edit control（确定密码））

显示“两次输入密码不一致”通知框

if ( 账号未曾注册且两次密码一致 )

将账号密码存入users数据库

XXX.DoModal(); //跳转至XXX（登陆）对话框

}

**返回**

{

XXX.DoModal(); //跳转至XXX（登陆）对话框

}

**注册界面示意图如下所示**

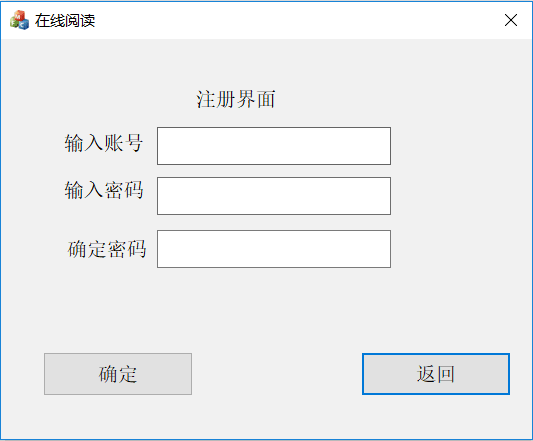


图 3- 2 注册界面

**三、主界面UI功能设计**

建立方式同登陆界面，控件布置包括：两个Static Text静态文本框分别显示文本 “书籍目录”、“我的收藏” ； 两个Combo Box组合框放置文本“书籍目录”、“我的收藏”后作为用户登陆选择书籍区间 ； 两个Button按钮分别是“跳转”以及“退出”。

完成控件布置后，为两个Combo Box以及两个Button添加变量

“书籍目录”——combo1

“我的收藏”——combo2

“跳转”——button6

“退出”——button6

为各个combo添加响应事件OnCbnSetfocus以及OnCbnSelchange

再为各个button添加响应事件OnBnClicked（）

各自伪代码如下所示

**书籍目录**

OnCbnSetfocus combo1

{

连接数据库books

遍历数据库

m\_cbWebs.AddString(\_T("books bname")); //将书名作为combo box中的项显示

}

OnCbnSelchange combo1

{

//检测combo box中选择的选项变化后调用

int index == 当前combo box中项的序列号

通过序列号得到字符串（书名）存储进一个全局变量Bookname中等待button调用。

}

**我的收藏目录**

OnCbnSetfocus combo2

{

连接数据库collect

遍历数据库

m\_cbWebs.AddString(\_T("collect cname")); //将书名作为combo box中的项显示

}

OnCbnSelchange combo2

{

//检测combo box中选择的选项变化后调用

int index == 当前combo box中项的序列号

通过序列号得到字符串（书名）存储进一个全局变量Collectname中等待button调用。

}

**跳转**

{

if（Bookname！=初始值&&Collectname！=初始值&& Bookname！= Collectname）

显示“不能同时选择两本书”

if(只有一个书籍名全局变量不为初始值)

将其赋给新全局变量truename；并再将对应页号赋值给一个整型全局变量page（if判定中如果是收藏数据库就将ccp赋予；如果if中判定是书籍数据库就将1（第一页）赋予）

XXX.DoModal(); //跳转至XXX（阅读）对话框

}

**退出**

{

OnCancel();

}

**主页面示意图如下所示**

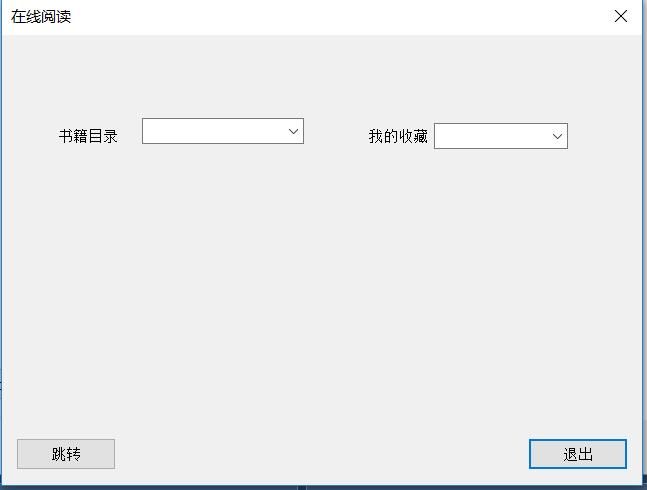


图 3- 3 主页面

**四、书籍阅读界面UI功能设计**

建立方式同登陆界面，控件布置包括： 一个Edit Control编辑框作为文本载体； 五个Button按钮分别是“显示”，“上一页”、“下一页”、“收藏”以及“返回”。

完成控件布置后，为三个Button添加变量

“显示”——button8

“上一页”——button9

“下一页”——button10

“收藏”——button11

“返回”——button12

再为各个button添加响应事件OnBnClicked（）

各自伪代码如下所示

**显示（最开始的初始化显示）**

{

根据全局变量得到书籍名以及对应的页数

连接数据库books

GetWindowText(Cstring)显示books对应page的txt

当前page赋给新的全局变量int currentp

}

**上一页**

{

根据全局变量得到书籍名以及对应的数据库

连接数据库books

if (current<=0)

显示“没有上一页”通知栏

else

currentp - =currentp;

GetWindowText(Cstring)显示books对应page的txt

}

**下一页**

{

根据全局变量得到书籍名以及对应的数据库

连接数据库books

查询数据库books最大的页数maxbcp

if (current+1 > maxbcp)

显示“没有下一页”通知栏

else

currentp + =currentp;

GetWindowText(Cstring)显示books对应page的txt

}

**收藏**

{

连接数据库collect

将全局变量logid（登陆操作时保存的）、currentp以及truename记录进collect

}

**返回**

{

XXX.DoModal(); //跳转至XXX（主页面）对话框

}

**书籍阅读界面如下所示**

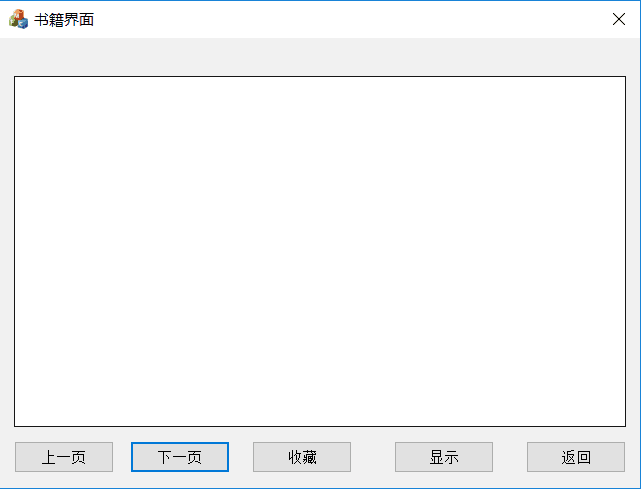


图 3- 4 书籍阅读界面

**通知栏模板示意图如下**

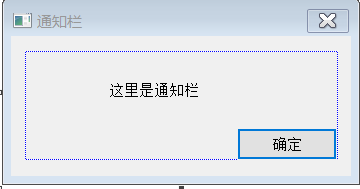


图 3- 5 通知栏模板

# 四.小组分工

课程设计主要工作我们分成四个模块

1. Socket套接字模块，主要负责Client,AS,TGS,Server四个端之间的Socket套接字通信函数，能确保四个端之间正常通信。这一部分由刘会禹负责。
2. 数据包的封装加密解封装等操作，主要负责将数据整合成按统一标准规定的数据包以及将数据包解封装为各种数据，并负责加密解密操作。这一部分由邹鑫娣负责。
3. UI设计以及与与UI前端操作相关的功能函数设计。这一部分由罗一飞负责
4. 数据库管理，负责数据库的连接，管理以及与数据库相关的查询函数设计。这一部分由贾晨阳负责。