

《物联网安全》

设计文档

组 长： 魏来 班号： 193182

组 员： 郭鸿绅、 段依琳、 刘畅海

院（系）： 计算机学院 专业： 网络工程

指导教师： 姚 宏 职称： 教授

2021 年 5 月

目录

[《物联网安全》 1](#_Toc16391)

[设计文档 1](#_Toc5483)

[1. 序言 1](#_Toc3099)

[1.1. 程序目的 1](#_Toc29116)

[1.2. 定义 1](#_Toc6800)

[2. 需求分析 2](#_Toc12700)

[2.1. 任务概述 2](#_Toc7118)

[2.2. 数据流图 4](#_Toc25443)

[2.2.1. 顶层数据流图 4](#_Toc32151)

[2.2.2. 一层数据流图 4](#_Toc4564)

[2.2.3. P1二层数据流图 5](#_Toc19324)

[2.2.4. P2二层数据流图 5](#_Toc7832)

[2.2.5. P3二层数据流图 6](#_Toc15229)

[2.3. 状态机设计 6](#_Toc7895)

[2.4. UI布局设计 7](#_Toc4251)

[3. 系统概要设计 12](#_Toc18951)

[3.1. 系统概述 12](#_Toc2827)

[3.1.1功能需求规定： 12](#_Toc29653)

[3.1.2运行环境： 12](#_Toc6890)

[3.2. 系统结构 12](#_Toc4738)

[3.3. 模块设计 13](#_Toc29088)

*[4. 数据](#_Toc7491)* [14](#_Toc7491)

[4.1. 时序图 14](#_Toc12119)

[4.1.1. 证书时序图 14](#_Toc10834)

[4.1.2. Kerberos认证时序图 15](#_Toc5424)

[4.1.3. 应用时序图 16](#_Toc10783)

[4.2. 数据报文 17](#_Toc21195)

[4.2.1. K协议 17](#_Toc28557)

[4.2.2. EC协议 18](#_Toc19877)

[4.2.3. WEC协议 22](#_Toc27703)

[5. 详细设计 23](#_Toc9252)

[5.1. 详细流程图 23](#_Toc18149)

[5.2. 数据结构 26](#_Toc11959)

[5.3. 模块详细设计 26](#_Toc14315)

[5.4. 函数设计 26](#_Toc1925)

[5.5. 函数流程图 26](#_Toc3980)

[5.6. 甘特图 26](#_Toc10659)

# 序言

## 1.1. 程序目的

在学习了对称加密—DES，和非对称加密—RSA之后，目前，课程设计需要将上述两种工具，以及Kerberos的认证体系结合，并应用于自定义的网络应用中。

在近期的讨论中，我们小组认为本次课程设计的重点并不在于Kerberos认证过程，而是实现一套逻辑严密，结构完整，内容丰富的网络应用，并在应用的使用过程中体现出Kerberos认证过程以及两套加解密内容；经过小组讨论并结合了现有水平能力，时间限制后，我们小组从简单的网络游戏，模拟美团点餐系统，IM（即时通讯）中敲定了本次课程设计的大致方向，即设计一套能够满足一定量并发需求的C/S模式的通讯应用，通过端到端的通讯过程，体现Kerberos认证流程与RSA加密过程，在通讯内容中体现DES加密过程。

本次应用设计中，我们将尽可能将程序模块拆开掰碎，使得模块分工细致完整，同时降低函数间，模块间的耦合度，提高程序的可拓展性；同时在实现基础通讯功能外，还构想了邀约游戏等功能，可以理解为一套新的程序，作为一个黑盒子添加到通讯程序的拓展模块，使得程序内容更加丰富。

魏来

2021年4月28日

## 1.2. 定义

|  |  |
| --- | --- |
| **专业术语** | **定义** |
| **Kerberos** | 计算机认证协议 |
| **数据库** | 用于存放数据，租借的阿里云MySQL数据库  rm-uf6t4cbyfz681x569.mysql.rds.aliyuncs.com 端口：3306 |
| **终端** | 4台设备模拟启动6个终端，即6个设备，分为KDC和User两种类别 |
| **KDC** | 密钥分配中心，由两个终端扮演，在本次程序中分为AS终端和TGS终端 |
| **AS** | 身份验证服务，为client生成TGT的服务 |
| **TGS** | 票务发放服务，为client生成某个服务的ticket |
| **TGT** | 用于获取ticket的票据 |
| **ticket** | 用于发送信息的凭证 |
| **User** | 实际运行应用程序的设备，又分为Client，Server |
| **Server** | 通讯服务的服务器设备 |
| **Client** | 需要通讯的用户的设备 |
| **K（）** | 密钥，用于加解密工具 |
| **聊天用户** | 使用本程序的用户 |
| **Session key** | 发送应用信息时，DES加密时用到的秘钥 |

表 1 名词定义

|  |  |
| --- | --- |
| **密钥表** | **意义** |
|  | User证书中的的公钥，用于发送对称钥和数字签名 |
|  | 对称钥，由AS生成，通过加密发送给User |
|  | TGS和另一个User之间的对称钥，与等价 |
|  | c和TGS之间的对称钥，用于发送session key |
|  | c和v之间的session key |
|  | AS和TGS之间的对称钥，写入文本。 |

表 2 密钥定义

# 需求分析

## 任务概述

本程序为基于 Kerberos 认证（DES加密）和 RSA加密的分布式通讯工具，需要实现下述部分：

工具：

1. 加密工具：使用DES与RSA两种加密方式，根据报文的不同需求，分别使用不同的加密方式。
2. RSA加密中的证书：约定好证书格式后，每个设备启动时，都会生成自己的证书，同时我们认为这个证书在系统内，独一无二，绝对真实，绝对安全，因此，先通过明文通讯，使得所有User设备与KDC设备间得到对方的公钥，User中，Client与Server得到对方公钥，即证书互信。同时可以认为系统内的Client与Server间都有对方的公钥，传输后续数组签名用到的密钥。
3. 数字签名，通过2）中获得的公钥，并用RSA给每条数据报文都加上数字签名。

Kerberos认证：

1. 在Kerberos认证之前：由于Kerberos本身无RSA加解密需求，但为了将该功能引入Kerberos中，我们将最开始认证中出现的作为明文，根据最初已经获得的证书信息，使用RSA加密，通过通信功能传输。
2. Kerberos认证体系：当所有设备启动后，其中两台设备扮演TGS和AS的角色，一台设备扮演通讯工具中的Server，其他设备扮演Client；所有Client根据自己在AS中的ID信息与Server执行Kerberos认证，AS和TGS为每一个Client与Server间提供认证服务（由Client发起），Client与Client间没有必要提供认证服务。
3. 通过上述顺序为4）---> 5）--->2）的逻辑，实现了Kerberos认证功能后，认为除了出现新的User设备，整个系统内无需再使用Kerberos认证，AS和TGS设备监听新设备，不执行后续逻辑功能。

应用：

①客户端：提供登录、主窗体及聊天等界面及对应的业务逻辑，向服务器发送相应的服务请求，并接受相应的处理结果。客户端是轻量级的软件，只负责链接远程服务器，并发出相应的服务请求，并不进行核心业务逻辑的处理。具体的处理交给服务器，而客户端只接收服务器处理的结果并显示给用户。

②服务器：监控登录信息及在线用户信息，接收客户端的服务请求，并做相应的处理，然后将处理结果发送给客户端。服务器负责处理核心的业务逻辑，并负责连接数据库，保存和读取数据。因此，服务器端设计的好坏也直接影响即时通信软件的质量

用例描述：



图 1 用户（服务器）用例图

## 数据流图

### 顶层数据流图



图 2 顶层数据流图

### 一层数据流图



图 3 一层数据流图

### P1二层数据流图



图 4 P1二层数据流图

### P2二层数据流图



图 5 P2二层数据流图

### P3二层数据流图



图 6 P3二层数据流图

## 状态机设计



图 7 Server状态机

## UI布局设计

注：初始阶段，没有通过WPF控件等工具制作UI，采用手绘的方式大致分析UI界面，后期会对UI界面进行美化处理，但大体布局不变。

登录界面

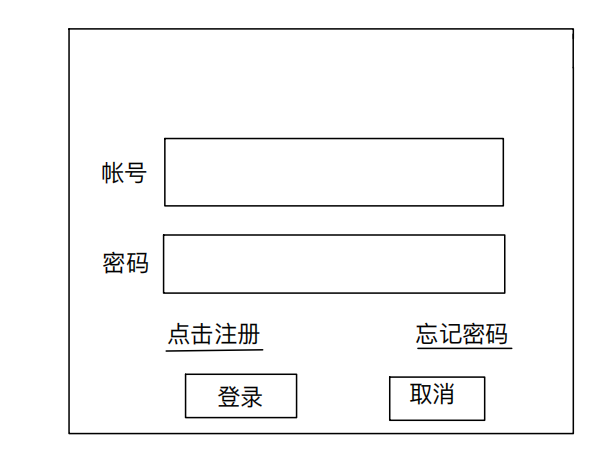


图 8 登录界面

1. 消息列表

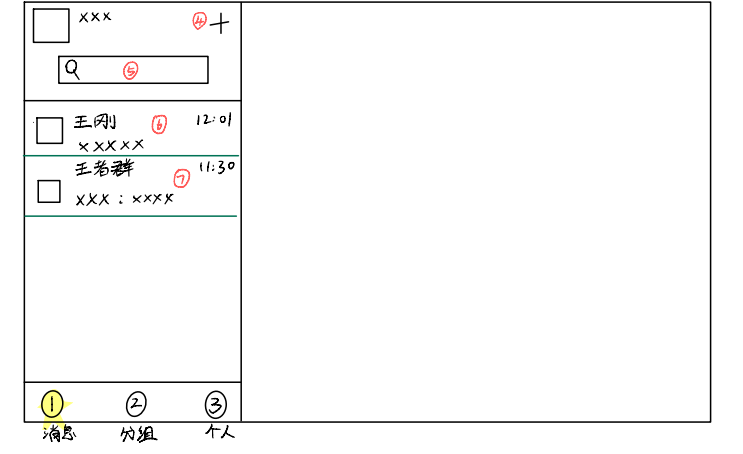


图 9 消息列表界面

1. 查看联系人界面

8与王刚聊天，右侧跳转至6，9跳转至7

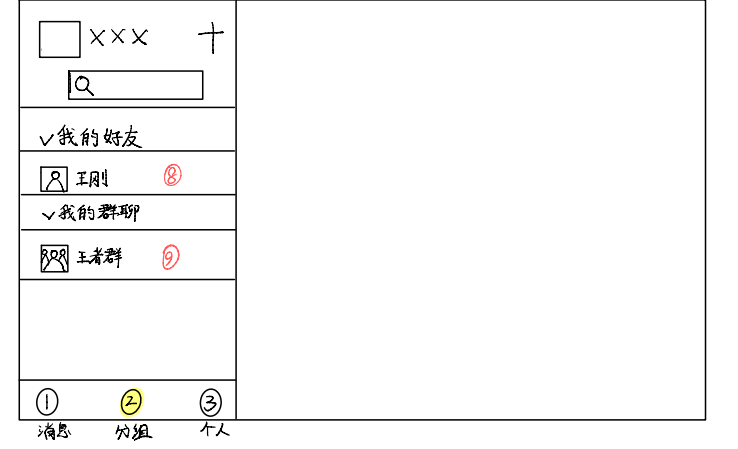


图 10 查看联系人界面

1. 查看个人信息界面

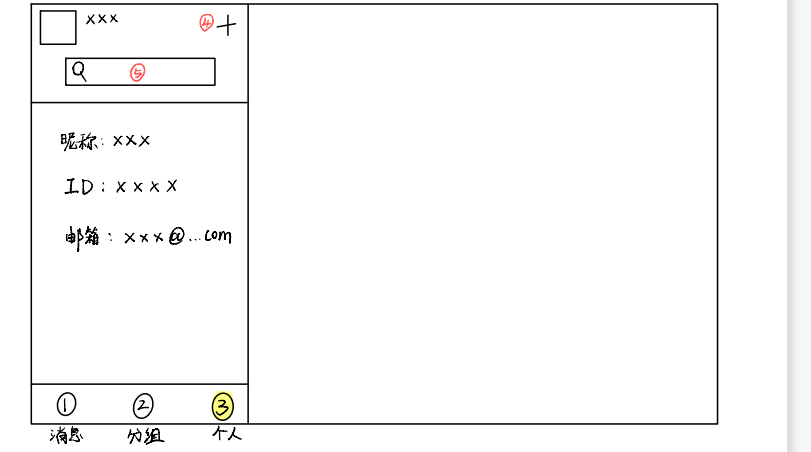


图 11 查看个人信息界面

1. 添加界面

点击+号，弹出功能

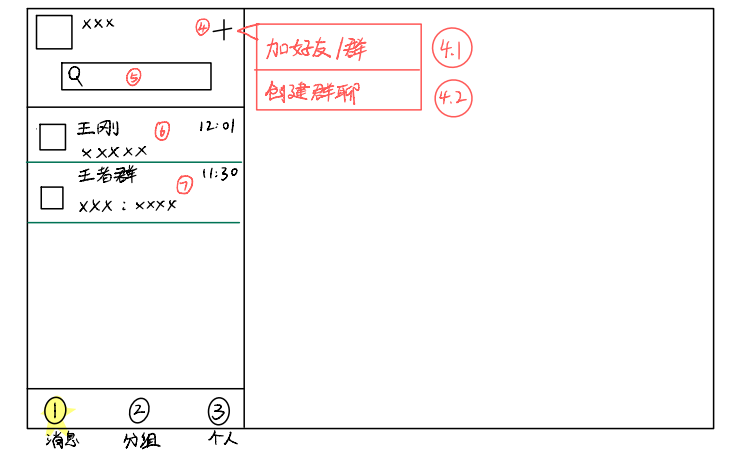


图 12 添加界面

* 1. 添加好友、群界面

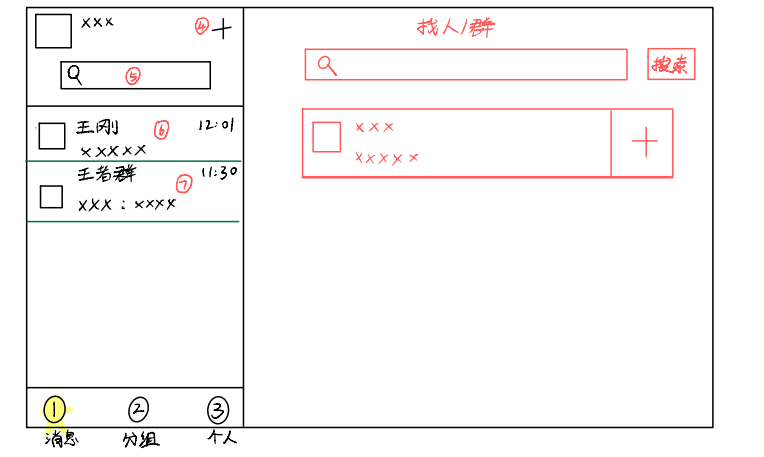


图 13 添加好友、群聊界面

* 1. 创建群聊界面

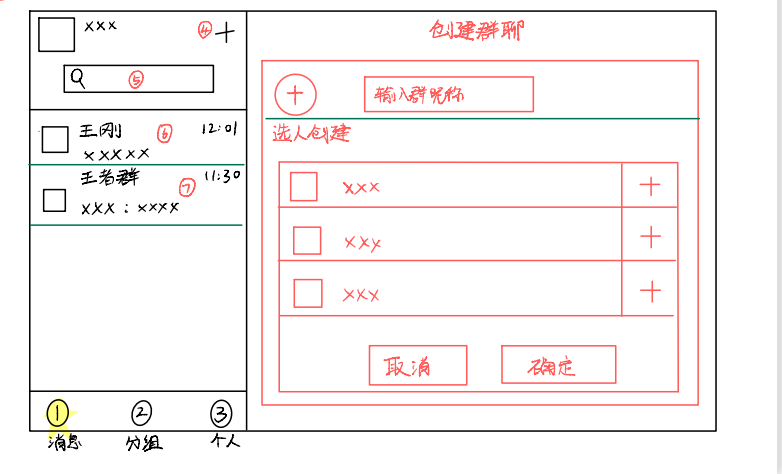


图 14 创建群聊界面

1. 查看信息界面

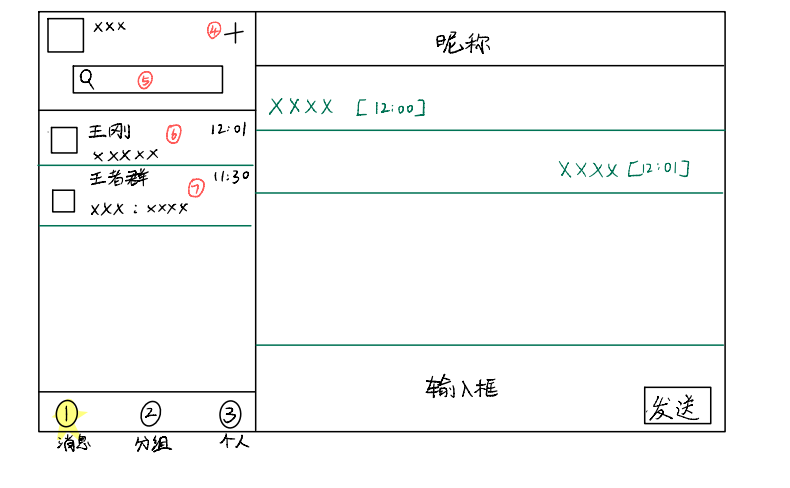


图 15 查看信息界面

1. 查看群信息界面

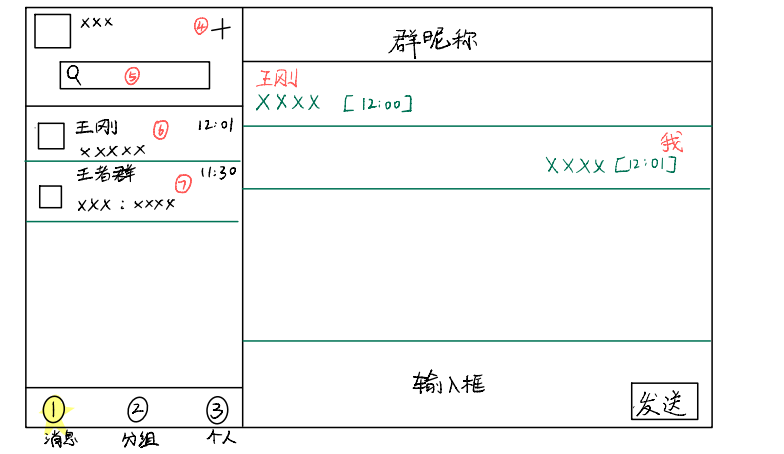


图 16 查看群信息界面

# 系统概要设计

## 系统概述

### 3.1.1功能需求规定：

AS系统负责，接收User传过来的证书以及向User端发放TGT；

TGS系统验证User的TGT，生成并发放密钥。

Client系统完成Kerberos系统认证：向AS请求TGT，凭TGT向TGS请求密钥，与Server系统完成双向验证；完成与Server端的交互，包括，聊天，添加好友、群，删除好友，退出群聊，登录，注册，查看个人信息。

Server系统完成Client系统的交互请求，转发信息。

### 3.1.2运行环境：

运行平台：Windows10（AS、TGS、Client），Linux（Server）

数据库：MySQL

Web：Node.js平台下的JavaScript

开发语言：C#、Java、JavaScript

## 系统结构



图 17 系统结构流程图

## 功能模块设计



图 18 AS功能图



图 19 TGS功能图



图 20 Server功能图



图 21 Client功能图

## 拓扑结构设计

**

图 22 拓扑图

# *数据*

## 数据库设计

### 概念结构设计

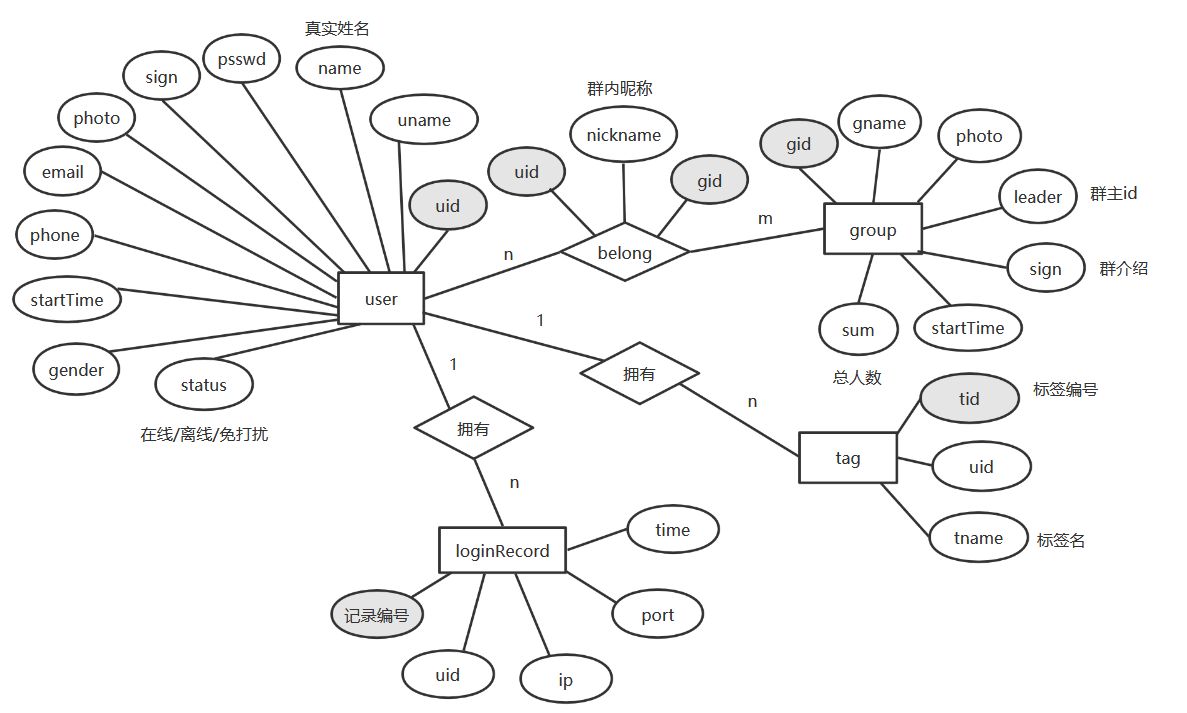


图 23 Server-ER图

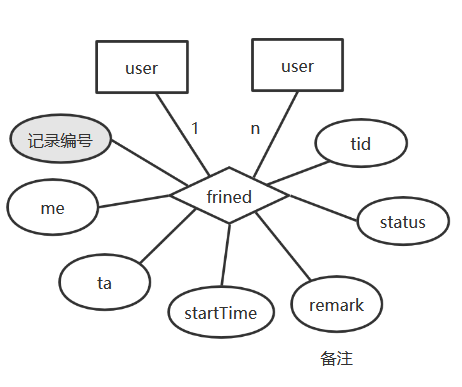


图 24 朋友表ER图

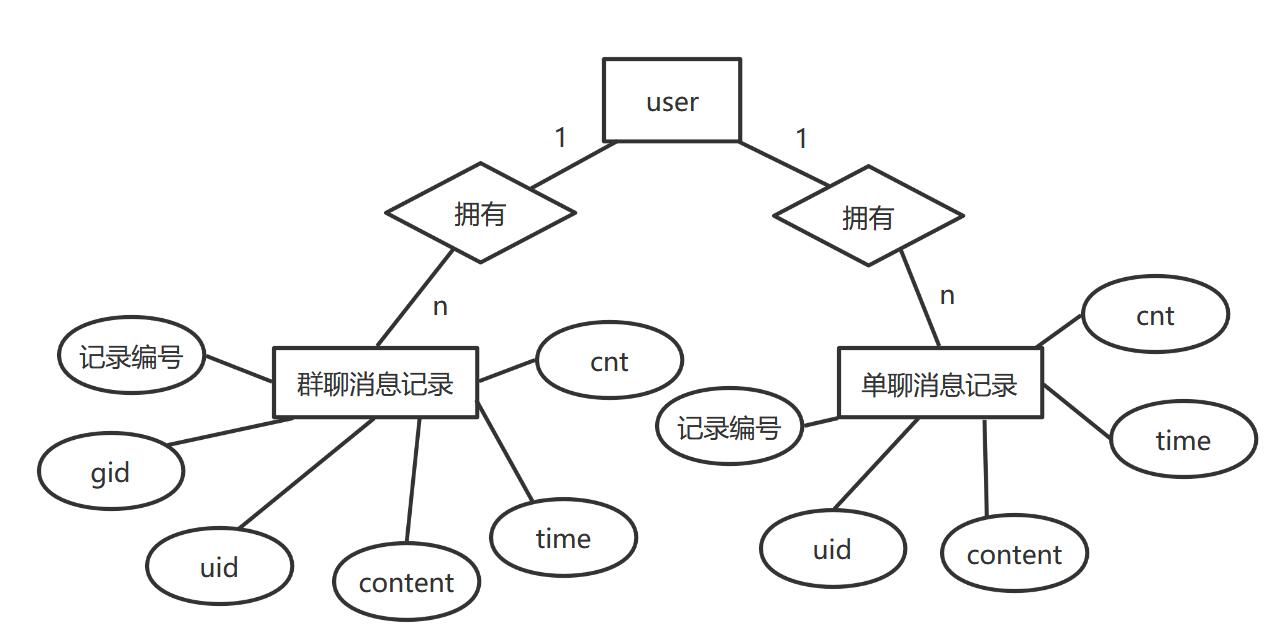


图 25 聊天ER图

### 表结构设计

users表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名称** | **含义** | **数据类型** | **是否可空** | **约束** |
| uid | 账号 | char(10) | 否 | 主键 |
| uname | 用户名 | nvarchar(20) | 否 |  |
| name | 真实姓名 | nvarchar(20) |  |  |
| psswd | 密码 | nvarchar(20) | 否 |  |
| sign | 个签 | nvarchar(50) |  |  |
| photo | 头像 | nvarchar(200) |  |  |
| email | 邮箱 | nvarchar(50) | 否 |  |
| phone | 电话 | varchar(20) |  |  |
| gender | 性别 | int |  |  |
| status | 状态 | int | 否 |  |
| startTime | 注册时间 | int | 否 |  |

friend表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名称** | **含义** | **数据类型** | **是否可空** | **约束** |
| 记录编号 | 编号 | int | 否 | 主键 |
| me | 用户账号 | char(10) | 否 |  |
| ta | 用户账号 | char(10) | 否 |  |
| startTime | 时间 | int | 否 |  |
| remark | 备注 | nvarchar(20) |  |  |
| tid | 标签编号 | int |  |  |
| status | 状态 | int |  |  |

groups表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名称** | **含义** | **数据类型** | **是否可空** | **约束** |
| gid | 群聊账号 | char(10) | 否 | 主键 |
| gname | 群聊名 | nvarchar(20) | 否 |  |
| photo | 头像 | nvarchar(200) |  |  |
| leader | 群主 | char(10) | 否 |  |
| sign | 群介绍 | nvarchar(50) |  |  |
| startTime | 创建时间 | int | 否 |  |
| sum | 人数 | int | 否 |  |

belong表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名称** | **含义** | **数据类型** | **是否可空** | **约束** |
| gid | 群聊账号 | char(10) | 否 | 外键  主键 |
| uid | 用户账号 | char(10) | 否 |
| nickname | 群内昵称 | nvarchar(20) |  |  |

loginRecords表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名称** | **含义** | **数据类型** | **是否可空** | **约束** |
| 记录编号 | 编号 | int | 否 | 主键 |
| uid | 用户账号 | char(10) | 否 |  |
| ip | IP地址 | nvarchar(50) | 否 |  |
| port | 端口 | int | 否 |  |
| time | 时间戳 | int | 否 |  |

tags表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名称** | **含义** | **数据类型** | **是否可空** | **约束** |
| tid | 标签编号 | int | 否 | 主键 |
| uid | 用户账号 | char(10) | 否 |  |
| tname | 标签名 | nvarchar(20) | 否 |  |

群聊消息记录（本地文件）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名称** | **含义** | **数据类型** | **是否可空** | **约束** |
| 记录编号 | 编号 | int | 否 | 主键 |
| gid | 群聊账号 | char(10) | 否 |  |
| uid | 用户账号 | char(10) | 否 |  |
| content | 内容 | nvarchar(MAX) | 否 |  |
| time | 时间戳 | int | 否 |  |

单聊消息记录（本地文件）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名称** | **含义** | **数据类型** | **是否可空** | **约束** |
| 记录编号 | 编号 | int | 否 | 主键 |
| uid | 用户账号 | char(10) | 否 |  |
| content | 内容 | nvarchar(MAX) | 否 |  |
| time | 时间戳 | int | 否 |  |

证书表（AS数据库）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名称** | **含义** | **数据类型** | **是否可空** | **约束** |
| version | 版本号 | nvarchar(20) | 否 |  |
| serial | 序列号 | nvarchar(20) | 否 |  |
| deadline | 有效日期 | nvarchar(20) | 否 |  |
| name | 主体名 | nvarchar(20) | 否 | 主键 |
| pk | 公钥 | nvarchar(20) | 否 |  |

## 时序图

### 证书时序图

在Kerberos认证之前：证书发送，获取公钥。

User向AS发送自己的证书，即自己的公钥，AS通过该公钥加密一个随机数（Kc或Kv），作为Kerberos认证第一步的使用的Kc，以及User间，Client向Server转发使用的Kv。

Client与Server互发证书，使双方掌握自身公钥。



图 26 证书时序图

### Kerberos认证时序图

在应用程序之前：Kerberos认证，获取Session key

Client向AS发送请求，获取TGT，AS收到请求后，通过Kc加密生成票据；

Client凭票向TGS发送请求，获取Session key，TGS根据票据随机生成Session key，并用Kv（AS与Server间的Kc）加密，发送给Client。

Client将session转发给Server，Server回复，实现相互认证。

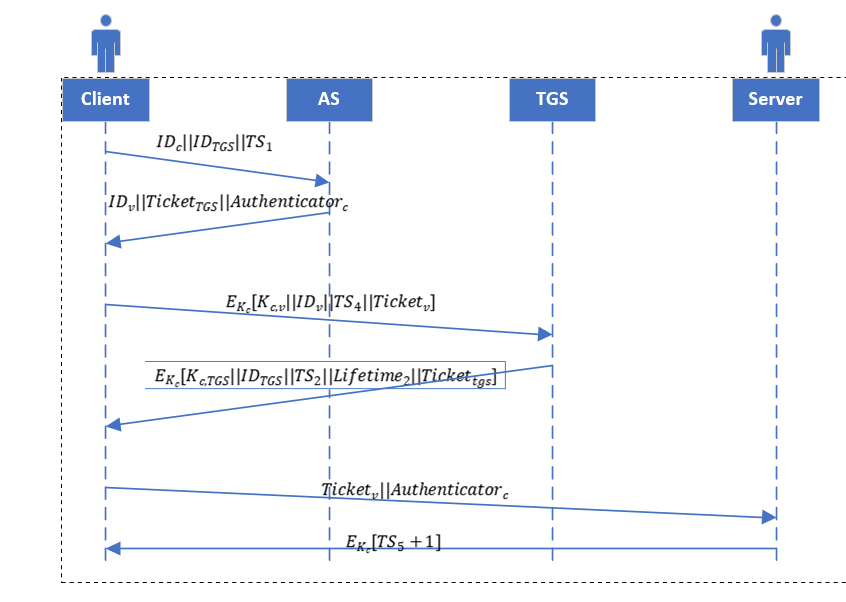


图 27 Kerberos认证时序图

### 应用时序图

应用程序：实现通讯

Client向Server发送注册请求，申请界面信息（好友群聊），Server给予响应，Client体现在UI中。



图 28 请求时序图

Client 登录，先向Server发送信息，Server回复Client登录信息，若登陆成功，Client向Server发送问好信息，Server拉取消息队列中未读的信息给Client。



图 29 登陆时序图

Client与Server建立的为长连接，因此，每隔一段之间，Client都要向Server发送一段心跳包，若Server在限制时间内未收到心跳包，则说明Client处于离线状态，更新数据库。



图 30 心跳包时序图

Client A向B发信息（添加好友、组建群等），先向Server发送信息，Server先放入消息暂存队列并打上时间戳，根据信息转发给Client B，若B不在线，则消息暂存队列不出队，若B收到信息，Client B将回复传达给Server，Server根据回复内容更新数据库，并转达给Client A，Client A、B均更新UI。



图 31 信息发送时序图

## 数据报文

Json是Java和C#之间通信的利器，Java端将Java对象转变为Json串后发出，C#端接收到Json串后转换为C#对象；C#发出转变为Json串的对象，Java收到后解析成Java对象，Json串在不同语言之间起到一个桥梁的作用。对定义的Java或C#对象生成Json字串，以及从Json字串生成Java或C#对象，有很方便的方法。

同时发送方根据Json也可以将复杂的Socket报文序列化，而收方简单的反序列化就可以提取报文，因此本次通信数据结构采用Json模块。

|  |  |  |
| --- | --- | --- |
| Json数据字段 | 内容描述 | 细分 |
| msgId | 消息唯一标识符 | |
| Src | 发送者id | |
| Dst | 接收者id | |
| msgType | 消息类型 | 0001 发送证书报文 |
| 0002 AS向User发送Kc |
| 0003 Kerberos(1) |
| 0004 Kerberos(2) |
| 0005 Kerberos(3) |
| 0006 Kerberos(4) |
| 0007 Kerberos(5) |
| 0008 Kerberos(6) |
| 1001 注册信息 |
| 1002 登录信息 |
| 1003 请求好友界面 |
| 1004 好友界面信息 |
| 1005 问好信息（拉取离线信息），同时好友上线提醒 |
| 1006 心跳信息 |
| 1007 查找信息 |
| 1008 登出信息（Server端登出），同时好友离线提醒 |
| 1009 个人信息 |
| 1010 修改个人信息 |
| 2001 单聊信息 |
| 2002 群聊信息 |
| extend | 拓展字段，理论上可以继续通过Json序列化数据类型 | |
| statusReport | 状态报告（0，1)，msgType+statusReport等价于ack | |
| sTS | 发送方时间戳 | |
| rTS | 接收方时间戳 | |
| ContentType | 消息内容类型（一般为冗余，当消息类型为单聊或群聊时有用） | 101 文本消息 |
| 102 图片消息 |
| 103 语音消息 |
| 104 表情消息 |
| 105 动图消息 |
| 9001 添加好友信息 |
| 9002 加入群聊信息 |
| 9003 组建群聊信息 |
| 9004 退出群聊信息 |
| 9005 删除好友信息 |
| 9006 同意加好友 |
| 9007 拒绝加好友 |
| 9008同意加群 |
| 9009拒绝加群 |
| Xxx 其他消息类型 |

表 3 数据报文

# 详细设计

## 数据结构

报文数据结构，即Json数据结构：

|  |  |  |
| --- | --- | --- |
| 结构名 | 内容 | 功能 |
| Json | String msgId | 消息唯一标识符 |
| String Src | 发送者id |
| String Dst | 接收者id |
| String msgType | 消息类型 |
| String extend | 拓展字段，理论上可以继续通过Json序列化数据类型 |
| bool statusReport | 状态报告（0，1)，msgType+statusReport等价于ack |
| String sTS | 发送方时间戳 |
| String rTS | 接收方时间戳 |
| String ContentType | 消息内容类型（一般为冗余，当消息类型为单聊或群聊时有用） |

表 4 Json数据结构

程序内部数据结构：

|  |  |  |  |
| --- | --- | --- | --- |
| 报文 | 结构名 | 内容 | 功能 |
| 证书 | certificate | String version | 版本号 |
| String serial | 序列号 |
| String deadline | 有效日期 |
| String name | 主体名 |
| Long long pk | 公钥 |
| 密钥 | My\_k | String | 对称钥 |
| Client请求tgt | Message1 | String | Client端用户标识符 |
| String | 请求访问tgs |
| String | 验证时间同步 |
| 返回tgt | Message2 | String | Session key副本 |
| String | 确认ticket为TGS制作 |
| String | 签发时间 |
| String | 有效期 |
| Class | 访问tgs的ticket |
| 票据 |  | String | Session key副本 |
| String | Client端用户标识符 |
| String | Client地址 |
| String | 确认ticket为TGS制作 |
| String | 签发时间 |
| String | 有效期 |
| 请求Server ticket | Message3 | String | 用户访问服务器v的id |
| Class | 向TGS证实自己被认证 |
| Class | 验证ticket |
| 返回Server ticket | Message4 | String | TGS生成session key |
| String | 用户访问服务器v的id |
| String | 签发时间 |
| Class | Client访问server的ticket |
| 验证ticket | Class | String | Ticket的主人 |
| String | Ticket的主人地址 |
| String | 签发时间 |
| Client请求服务 | Message5 | Class | 向服务器证实被认证 |
| Class | 验证ticket有效 |
| 客户对服务器的可选认证 | Message6 | String +1 | 确认不是报文重放 |
| 注册信息、登录信息 | User | String uid | 账号 |
| String unmae | 用户名 |
| String name | 真实姓名 |
| String psswd | 密码 |
| String sign | 个签 |
| String photo | 头像 |
| String email | 邮箱 |
| String phone | 电话 |
| Int gender | 性别 |
| Int status | 状态 |
| Int startTime | 注册时间（以1970年标准时间为基准） |
| 好友信息 | Friend | Class User | 用户结构 |
| String remark | 备注 |
| Int startTime | 加好友时间 |
| String tid | 分组 |
| 聊天信息 | chat\_Message | Int head | 信息种类 |
| String content | 信息内容 |
| Class User | 发送方 |
| Int time | 时间戳 |
| 群 | Group | List map[User,nick] | 用户列表 |
| String gid | 群账号 |
| String photo | 群头像 |
| String leader | 群主账号 |
| String gign | 群介绍 |
| Int startTime | 创建时间 |
| 群消息记录 |  | String gid | 群聊账号 |
| String uid | 用户账号 |
| String content | 内容 |
| Int time | 时间戳 |
| 单聊记录 |  | String uid | 用户账号 |
| String content | 内容 |
| Int time | 时间戳 |

## 详细流程图

流程描述：AS只能接收数据报文首部为0001的证书信息，或者为0003的Kerberos认证信息：

当接收到0001的报文时，通过证书结构体反序列化Json，可以得到证书信息，从证书中提取公钥pk，并利用RSA加密将随机生成的Kc加密发送给Client，Kc存入AS数据库。

当接收到0003的报文时，通过Kerberos1结构体反序列化Json，得到，然后根据上文提到的对称钥Kc加密，并将Json报文发送给Client。



图 32 AS流程图

流程描述：TGS只能接收数据报文首部为0005的Kerberos认证信息

当接收到0005的报文时，通过Kerberos3结构体反序列化Json，，AS与TGS共享数据库，TGS根据凭据，利用提前在AS与TGS提前约定好的Kc,v解密，验证身份，成功后，利用Kc加密并将Json报文转发给Client，若失败则发送ack信息。



图 33 TGS流程图

流程描述：Server有三种状态：等待连接状态、等待报文状态、处理报文状态。

最开始Server处于等待连接状态，当线程池中有线程接收到socket长连接后，改变状态进入等待报文状态，若此过程中，长期未收到心跳报文，则回到等待连接状态，若收到其他报文则进入报文处理状态，报文处理状态时，若收到了登出信息则跳转至等待连接状态。

当接收到0001的报文时，Server解析出证书信息，得到了证书主体对应的Pk，用于后续数字签名，同时将自己的证书发送给对应的Client，实现证书互信。

当接收到0007的报文时，Server根据认证模块中得到的Kv（即上述Kc），通过Des解密，得到。并根据中的时间戳，发送0008的报文，回复Client，完整Client与Server整个Kerberos认证模块。

当接收到1001，1010的注册，修改个人信息的报文，需要数据库的信息，会先验证数据库修改是否合法（如用户名冲、密码错误），然后返回ACK，若成功将状态为置为1，否则置为0，并返回给Client。

当接收到1005的问好信息，将数据库中用户的状态从离线转为上线状态，通知他的好友（即向好友转发问好信息），并将离线消息队列中的报文已2001单聊信息或者2002群里信息的格式发送给Client。

当长时间未接受心跳信息，或接收到登出信息时，修改数据库，并将等处信息通知他的好友，更新UI。

当接收报文为1002的登录信息时，Server根据内容验证，并返回ACK。

当接收报文为1003、1007、1009的搜索报文时，Server查询数据库，并将查寻到的列表通过Json序列号发送给Client。

当接受到群聊或单聊信息时，Server先将信息放入消息队列，由接收方是否上线的状态决定是否发送报文，然后根据消息内容类型判断报文为聊天信息，还是请求信息，执行不同逻辑。



图 34 Server流程图

流程描述：

Client端Kerberos认证，证书发送，Kc获取等与Server端代码无区别，因此没有重复体现。

当Client执行完认证等模块后，进入应用通讯程序，用户根据UI界面的状态选择功能。起始为登录界面，用户可以通过网页注册账号，或者在已有账号的状态下选择登录界面，与Server通讯，返回UI界面。

根据UI点击内容，与Server通讯不同的报文，执行聊天功能，如添加好友等。最后接收通过线程接收Server端报文，反馈UI。

可靠性考虑：

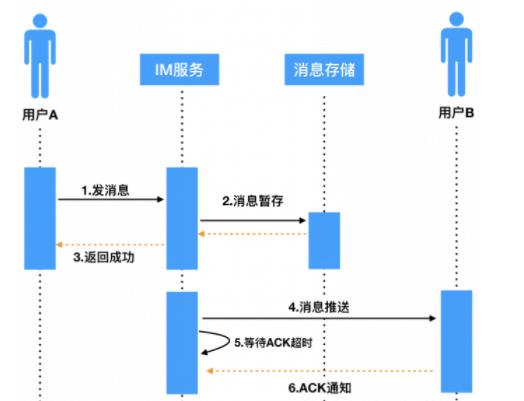


图 35 初步通讯流程时序图

参考上面时序图，发消息大概整体上分为两部分：

1.用户A发送消息到IM服务器，服务器将消息暂存，然后返回成功的结果给发送方A（步骤1,2,3）

2.IM服务器接着再将短暂的用户A发出的消息，推送给接收方用户B（步骤4）

其中可能丢失消息的场景有下面这些：

在第一部分中，步骤1,2,3都可能存在失败的情况。

由于用户A发消息时一个请求和响应的过程，如果用户A在把消息发送到IM服务器的过程中，由于网络不通等原因失败了；或者IM服务器接收到消息进行服务端存储时失败了；或者用户A等待IM服务器一定的超时时间，但IM服务器一直没有返回结果，那么这些情况用户A都会被提示发送失败。

接下来，他可以通过重试等方式来弥补，注意这里可能会导致发送重复消息的问题。

比如：客户端在超时时间内没有收到响应然后重试，但实际上，请求可能已经在服务端成功处理了，只是响应慢了，因此这种情况需要服务端有去重逻辑，一般发送端针对同一条重试消息有一个唯一的ID，便于服务端去重。

第二部分中。消息在IM服务器存储完后，响应用户A告知消息发送成功了，然后IM服务器把消息推送给用户B的在线设备。

在推送的准备阶段或者把消息写入到内核缓冲区后，如果服务端出现掉电，也会导致消息不能成功推送给用户B。这种情况实际上由于连接的IM服务器可能已经无法正常运转，需要通过后期的补救措施来解决丢消息的问题，后续详细介绍。

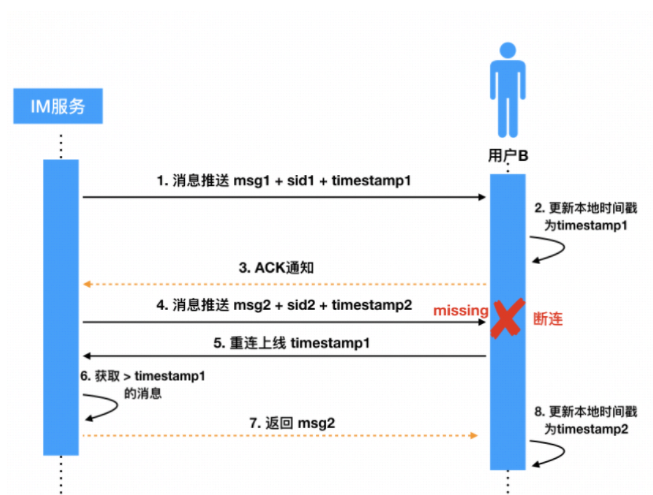
即使我们的消息成功通过TCP连接给到用户B的设备，但如果用户B的设备在接收后的处理过程出现问题，也会导致消息丢失。比如：用户B的设备在把消息写入本地DB时，出现异常导致没能成功入库，这种情况下，由于网络层面实际上已经成功投递了，但用户B却看不到消息。所以比较难处理。

解决方案：

1.针对第一部分，我们通过客户端A的超时重传和IM服务器的去重机制，基本就可以解决问题。

2.针对第二部分，业界一般参考TCP协议的ACK机制，实现一套业务层的ACK协议。

消息完整性检查



假设一台IM服务器在推送出消息后，由于硬件原因宕机了，这种情况下，如果这条消息真的丢了，由于负责的IM服务器宕机了无法触发重传，导致接收方B收不到这条消息。

问题在于：服务器机器宕机，重传这条路走不通了

那如果在用户B在重新上线时，让服务端有能力进行完整性检查，发现用户B有消息丢失的情况，就可以重新同步或者修复丢失的数据

比较常见的消息完整性检查的实现机制有时间戳比对

1、IM服务器给接收方B推送msg1，顺便带上一个最新的时间戳timestamp1，接收方B收到msg1后，更新本地最新消息的时间戳为timestamp1

2.IM服务器推送第二条消息msg2，带上一个当前最新的时间戳timestamp2，msg2在推送过程中由于某种原因接收B和IM服务器连接断开，导致msg2没有成功送达到接收方B。

3、用户B重新连上线，携带本地最新的时间戳timestamp1，IM服务器将用户B暂存的消息中时间戳大于timestamp1的所有消息返回给用户B，其中就包括之前没有成功的msg2.

4.用户b收到msg2后，更新本地最新消息的时间戳为timestamp2



图 36 Client流程图

## 模块详细设计

1. Client模块

2）AS、TGS模块

3）Server模块

4）工具模块（RSA、DES、数字签名）

5）通讯模块

|  |  |  |  |
| --- | --- | --- | --- |
| 接口名 | 参数 | 参数功能 | 函数功能 |
| My\_send | string a | IP地址 | 将数据报文c，通过socket通讯传输给IP为a，端口为b的终端 |
| stirng b | 端口号 |
| string c | 报文内容 |
| A | String s | 目的 id |  |
|  |  |  |  |
|  |  |  |  |

接口：My\_send(string a,stirng b,string c)

A,为ip，b为port，c为报文

## 函数设计

## 函数流程图

## 甘特图

Ex：具体函数代码，传入传出，有待商榷