注册过程

注册请求帧结构：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 注册标识 | 邀请码 | 用户名 | 密码 | 用户公钥 |

除了注册标识，其余字段都需要用server 公钥进行加密。

server公钥的配送有两种方式，一种是提前存储在客户端，这种方式灵活性差。另一种方法是当server收到一个连接，则发送自己的公钥给这个连接，这种方式有被中间人攻击的风险。暂且使用提前存储在客户端的方式。

邀请码由脚本生成，并存储到一个文件中，当server收到一个注册帧时，就会读取这个文件中的内容，匹配后就删除这个文件，可以通过是否删除这个文件来控制用户的注册。一个邀请码只使用一次。

用户名和密码同时存储于一个文件，其中密码用不可逆散列算法计算后存放，散列算法需要考虑加盐值来避免相同密码造成的散列值一样的情况。

服务器将用户公钥存放于一个文件，以后发送给该用户的所有消息都需要用这个公钥加密后才能发送。

注册回应帧：

|  |  |  |
| --- | --- | --- |
| 注册标识 | 注册状态 | 服务器分配的用户ID |

除了注册标识，其余字段在发送前都需要用用户发送的公钥进行加密。

注册状态有注册成功，注册失败等。

服务器分配的用户ID是服务器根据用户名经过散列算法计算得出，长度待定。相同的用户名会散列出相同的ID，不允许有相同的ID，所以也就不允许有相同的用户名。

注销过程

登陆过程

登陆帧结构：

|  |  |  |
| --- | --- | --- |
| 登陆标识 | 服务器分配的用户ID | 密码 |

除了注册标识，其余字段在发送前都需要用server 的公钥进行加密。

server收到登陆请求后，根据用户ID，将登陆帧中的密码用相同的散列算法计算，并与密码文件中的值匹配，匹配成功后更新用户状态为在线，并发送登陆结果回应帧给登陆请求用户。

登陆回应帧：

|  |  |
| --- | --- |
| 登陆标识 | 登陆结果 |

除了登陆标识，其余字段都需要用用户的公钥进行加密。

登陆结果包括登陆成功、登陆失败、重复登陆等。不允许重复登陆。

登出过程

添加好友过程

用户向其他人发送消息之前必须先添加对方为好友。

请求用户 -> server

|  |  |  |
| --- | --- | --- |
| 添加好友标识 | 请求用户ID | 目标用户名 |

请求用户首先向服务器发送添加好友的请求帧。服务器收到后先检查请求用户是否存在，不存在则不做任何处理。

server -> user2

|  |  |  |
| --- | --- | --- |
| 添加好友标识 | 请求用户名ID | 请求用户的公钥 |

服务器检查被添加的用户名是否已经注册，如果已经注册就直接发送添加请求帧，没有则回复请求用户，目标用户不存在。

user2 -> server

|  |  |  |
| --- | --- | --- |
| 添加好友标识 | 请求用户ID | 请求结果 |

用户2发送请求结果给server。

server -> user1

|  |  |  |  |
| --- | --- | --- | --- |
| 添加好友标识 | 被请求用户ID | 请求结果 | 被请求用户公钥 |

server发送请求结果给用户1，根据请求结果决定是否发送用户2的公钥。

会话过程

除了会话标识，其余字段都要用临时的AES加密。

当一个用户发起聊天时，server会创建一个会话，会话包括通信的用户。分为一对一，多对多两种情况。

一对一情况：

|  |  |  |  |
| --- | --- | --- | --- |
| 会话标识 | 发送用户ID | 接收用户ID | 消息 |

发送用户ID和接收用户ID字段部分用server的公钥加密，这是因为server的处理需要依赖这两个字段。数据部分用目的用户的公钥加密，由于是端到端加密，服务器不会知道用户发送的数据内容。

服务器根据发送用户ID判断该用户是否登陆，只有已经登陆的用户才能发送消息，如果该用户没有登陆则不做任何处理。服务器根据接收用户ID检查用户是否在线，如果不在线就将消息推入用户的消息队列，当用户上线是检查该队列。