架构

用户本地

- 整体设计参考p2p协议和icq等分布式协议，但是不设置中央服务器，设计以极致的反监视和隐私保护为目标，前端不考虑用户友好和UI设计，可以做成命令行
- 数据
  - 用户ID和昵称
  - 自己的公钥和私钥
  - 消息
- 密钥生成 — RSA — 8192位 — 生成公钥/私钥 — 公钥签名和公钥MD5
- 应用外的密钥储存
  - 字符串
    - 设置一个密码
    - 用这个密码加密RSA私钥（对称加密），并附上明文的MD5
    - 将加密过的私钥转换为32进制，生成一串包含字母和数字的字符串，交给用户要求其打印或手抄或储存于他认为安全的地方
    - 应用正常登录时中可以被再次读取生成，但是要验证密码
  - 二维码
    - 设置一个密码
    - 用这个密码加密RSA私钥（对称加密），并附上明文的MD5
    - 将加密过的私钥制作成3个二维码，每个二维码包含顺序信息和一部分的密钥
    - 要求用户打印二维码或者将其存于可信位置
    - 应用正常登录时中可以被再次读取生成，但是要验证密码
  - 加密狗 — 用加密狗密钥代替用户输入的密码或同时存在，剩余步骤相同
  - 文件 — 同样要求密码，然后直接将文件存于本地，要求用户妥善保存这个文件
- 登录验证
  - 输入字符串或者扫码或选择文件
  - 验证格式
  - 输入密码或/并插入加密狗
  - 解密
  - 验证MD5
- 发送
  - 添加时间戳，随机校验码（防止某些情况造成的重复收到消息），RSA公钥加密和MD5运算
  - 检查连接（握手）
    - 握手不成功
    - 从tracker获取连接
      - 判定对方未上线或者状态未知
      - 离线：暂存于本地等待发送
      - 不定：发送一份并暂存一份等待确认上线后发送
  - 握手成功
  - 发送密文和明文的MD5
- 接收
  - 握手成功
  - 接收密文和MD5
  - RSA私钥解密和MD5验证
    - MD5正确 — 就当无事发生
    - MD5不正确
      - 重新发送公钥并要求重新发送信息
      - 次数超过限制：“严重的连接错误”
  - 检查校验码和时间戳排除重复消息，根据时间戳进行消息顺序排列
- 群聊
  - 加群
    - 搜索
    - 验证（可选）
    - 加入
  - 消息
    - 1-50人
      - 类似Gossip协议
      - 发送
        - 判断在线状态，获取在线状态表
        - 建立发送路线表
          - fan-out=10（在线人数正常）
fan-out=50（在线人数少）
          - 除了最后一级其他必须为在线成员
          - 除了第一级，其他每级至少收到来自三个上一级的发送
          - 下一级有不在线成员的将不在线成员id和对应消息的时间戳储存于“待发送信息表中”
          - 最后一级可以包含不在线成员，在上线后广播全群，然后全员检索“待发送信息表”并发送未发送信息
        - 添加时间戳（ms级别精确值），发送路线表，随机校验码，发送者ID，下一级成员的RSA公钥加密和添加明文的MD5
        - 向下一级发送密文和明文的MD5
        - 下一级收到消息
        - RSA私钥解密和MD5验证
          - MD5正确 — 就当无事发生
          - MD5不正确 — 重新发送公钥并要求重新发送
        - 根据时间戳排列和显示消息，根据ID显示头像和昵称
        - 读取发送路线表
        - 判断是否需要再发送
        - 否 — 是 — 判断下一级成员是否全部在线
          - 是
          - 否
            - 在线成员
            - 不在线成员
              - 将对方ID和消息时间戳储存于待发送信息表内
              - 收到对方上线广播
    - 51-2500人
      - 类似gossip协议
      - 随机挑选数量为总人数的二次方根的用户，暂且称他们为零级成员 — 判断在线状态
        - 部分不在线 — 排除不在线的并重新随机挑，直到全部都是在线的。超过10轮依旧不是全部在线就直接获全群的在线状态，直接选出在线的人，如果数量达不到就直接强制下一步
        - 全部在线 — 继续
      - 建立发送路线图
        - 将剩下的用户进行分表
          - 表的数量=零级成员的数量
          - 每个用户至少被包含在3个表中
        - 在每个表内建立发送路线表
          - fan-out=10（在线人数正常）
fan-out=50（在线人数少）
（仅限非最后一级）
          - 除了最后一级其他必须为在线成员
          - 除了第一级，其他每级至少收到来自三个上一级的发送
          - 下一级有不在线成员的将不在线成员id和对应消息的时间戳储存于“待发送信息表中”
          - 最后一级可以包含不在线成员，在上线后广播全群，然后全员检索“待发送信息表”并发送未发送信息
      - 添加时间戳（ms级别精确值），每个零级成员的发送路线表，随机校验码，发送者ID，零级成员的RSA公钥加密和添加明文的MD5
      - 消息分发给每个零级成员
      - 零级成员对消息进行表内分发
        - 添加时间戳（ms级别精确值），发送路线表，随机校验码，发送者ID，下一级成员的RSA公钥加密和添加明文的MD5
        - 向下一级发送密文和明文的MD5
        - 下一级收到消息
        - RSA私钥解密和MD5验证
          - MD5正确 — 就当无事发生
          - MD5不正确 — 重新发送公钥并要求重新发送
        - 根据时间戳排列和显示消息
        - 读取发送路线表
        - 判断是否需要再发送
        - 否 — 是 — 判断下一级成员是否全部在线
          - 是
          - 否
            - 在线成员
            - 不在线成员
              - 将对方ID和消息时间戳储存于待发送信息表内
              - 收到对方上线广播
  - 群管理
    - 禁言
    - 撤回
    - 踢出群聊
- 高级身份验证
  - 用户间自己定一个密码
  - 身份怀疑时，发送验证
  - Digest Authentication
- tracker获取
  - 随版本更新推送
  - 用户手动添加
- 添加好友
  - 一般
    - 发现
      - 连接tracker查询ID
      - 获取连接
      - 连接对方，获取昵称，头像
    - 发送 — 发送ID，昵称，头像和签名的公钥以及公钥的MD5
    - 接受
      - 储存对方的ID和公钥
      - tracker查询建立连接
      - 发送签名的的公钥和MD5
  - 更高安全性
    - 加密密钥 — 密码的MD5
    - 发现 — 同上
    - 发送 — 信息同上，但是公钥用AES加密
    - 接受 — 填写密码，解密公钥部分，其余同上
  - 无tracker
    - 手动添加文件
    - 文件内容
      - ID 昵称，头像
      - 公钥
      - IP（公网或内网穿透） — 此项可手动更新
    - 添加 — 检测、建立连接 — 发送自己的信息和公钥
  - 带验证问题
    - 从服务器接用户信息和加密的联系方式和签名的验证问题明文
    - 回答问题，进行MD5
    - 使用MD5解密联系方式判断正误
- 在线状态查询
  - 状态检测
    - 发送测试信息（随机数）
    - 计算并返回（随机数+公钥， MD5）
    - 验证
      - 正常响应 — 判定在线
      - 无响应或者错误
        - 连接tracker更新连接方式和在线状态
        - 连接方式有更新，在线 — 重新执行状态检测
        - 连接方式有更新，离线 — 重新执行状态检测
          - 通过 — 在线
          - 不通过 — 离线
        - 连接方式无更新，离线 — 离线
        - 连接方式无更新，在线 — “状态未知”
  - 状态检测2 — Gossip传播的在线状态
  - 检测时机
    - 全局和用户信息更新点
    - 手动
- 信息更新点判定
  - 用户
    - 进入用户聊天界面时
    - 发送信息时
  - 全局
    - 进入应用时
    - 每隔一段时间（用户设定）（仅前台）
- 状态更新
  - 连接方式 — 系统端口获取网络状态变化，变化时检测网络并发送新的连接方式到tracker，在PPP连不上时也进行刷新
  - 在线状态
    - 手动下线时发送更新给tracker
    - 上线时发送更新给tracker，同时gossip
  - 昵称头像更新
    - 更新时Gossip传播（传播内容为「ID：xx头像更新」，不直接传播头像）
    - 进入聊天界面时传输头像昵称的MD5
      - 一致 — 不管
      - 不一致 — PPP传输更新
  - 切换设备登录
- 内容审查 — 无
- 应用内防火墙
  - 此部分用于防止第三方修改客户端导致的泄漏
  - 防止应用本身与非用户和tracker的URL/IP联系 — 通讯时检索IP和连接列表
  - 防止伪装成tracker或用户的信息跟踪服务器 — 检查流量格式和数量
  - 此部分要求不能被轻易删除或修改，通过某些玄学方式让整个应用的核心功能没这玩意了就跑不动
- http伪装 — 参考v2ray，对流量进行http形式的伪装并尽可能的模拟http服务器的行为
- 密钥的手动更新
  - 用户感觉自己的密钥被泄漏之后可以手动更新密钥
  - 进入某个聊天界面后向对方发送公钥

tracker

- 设计目标：可以被简单快速的部署（x-ui的部署难度）不储存隐私信息或可能导致严重隐私泄漏（聊天记录等泄漏）的信息，储存尽可能少的文件，尽可能低的cpu和带宽使用（要求在服务1000w用户时本地储存的信息不超过20GB，带宽不超过3mbps），带有简易防火墙（或者部署完给他们推荐个防火墙软件），可以抵挡一定程度的攻击，信息不能被服务器所有者直接查看。
- 储存信息：（对称加密）
连接到此处的用户ID
在线状态和通讯方式
以及验证问题 — 加密方式 — 用一个密钥对信息进行bitlocker，密钥在写软件的时候就定下，不放在开源库中，在软件中用字节码加密
- 用户查询
  - 接收用户发来的指令
  - 发送指令要求的信息
- 状态判定 — 接收客户端发来的状态更新
- 连接信息更新 — 等用户上线连上tracker把连接信息发过来

付款计划

- 完成“密钥生成”和“用户ID生成”（理想状态下ID生成是完全本地处理的，可能直接用ssid加一些算法即可？） — 300
- 完成无tracker下的“发送”与“接收”（即两个公网IP之间的RSA加密实时通讯） — 800
- 完成客户端的文件系统 — 300
- 客户端加入“联系人列表”和“消息记录” — 300
- 完成tracker的文件系统 — 600
- 完成tracker的“建立客户端之间端到端通讯”功能（参考P2P） — 3000
- 完成客户端与tracker的“在线状态更新”功能 — 2000
- 完善tracker — 5000
- 客户端接入tracker，完成“tracker获取”
- 完善新用户注册和登录（“登陆验证”“应用外密钥储存”） — 2000
- “高级身份验证” — 800
- 完成“好友添加”的“一般”和“无tracker” — 1500
- 完成“好友添加”的“更高安全性”和“带验证问题” — 2000
- 完成“在线状态查询” — 1800
- 完成“信息点更新判定” — 500
- 完成“http伪装” — 500
- 完成”应用内防火墙“ — 2500