基于交换密钥的 P2P 即时消息通讯和链上加密存储实现 - 设计文档

所在赛道与赛项：A-ST

一、 目标问题与意义价值

随着移动互联网和智能手机的快速发展，即时通讯软件（Instant Message）已经成为人们日常生活中必不可少的一部分。近年来，越来越多的数据安全问题也开始浮现出来。在传统聊天软件中，应用采用 server/client 的中心化模式，将用户敏感的聊天记录储存于中心数据库中，一旦被黑客攻击，就可以轻松窃取用户的隐私聊天内容。例如在 2018 年 Facebook 遭遇了大规模数据泄漏事件；2021 年 2 月，Facebook Messenger 的用户数据被黑客窃取，同年 WhatsApp 的用户信息和聊天记录被黑客出售；这些都表明了传统聊天软件所面临安全风险和挑战。

本软件提出一种基于 web3 技术的即时通讯软件。采用 web3 支持的 p2p 协议库连接对等点并使用非对称密钥加密通讯内容，基于 onebot v12 标准创建通信协议，并去中心化离线储存。同时实现基于 IPFS 的储存持久化，通过引入 ECDSA ECDH 交换密钥算法和国密 SM4 加密算法，为自身消息记录实现加密和哈希，设计 IPFS MFS 文件目录索引，最终上传至 IPFS 并发布至 IPNS。

本软件设计了一种快速、安全且完整的端到端即时通讯软件。在企业内部沟通方面，该软件可作为一种安全的交流工具，在政府机构等重要机密场所也是该聊天工具的应用场景；也可应用于个人的安全通信需求，如医生和患者的远程通信需求，可以有效保障患者的隐私。

二、 设计思路与方案

软件前端采用 Vue3 框架和 DaisyUI 组件库实现，利用 Axios 网络库与后端进行通信，并进行响应式数据更新。软件后端使用 Golang 实现，可划分为 HTTP 处理模块、P2P 连接模块、IPFS 交互模块、LevelDB 本地存储模块和加密模块。软件架构图如图 1 所示。

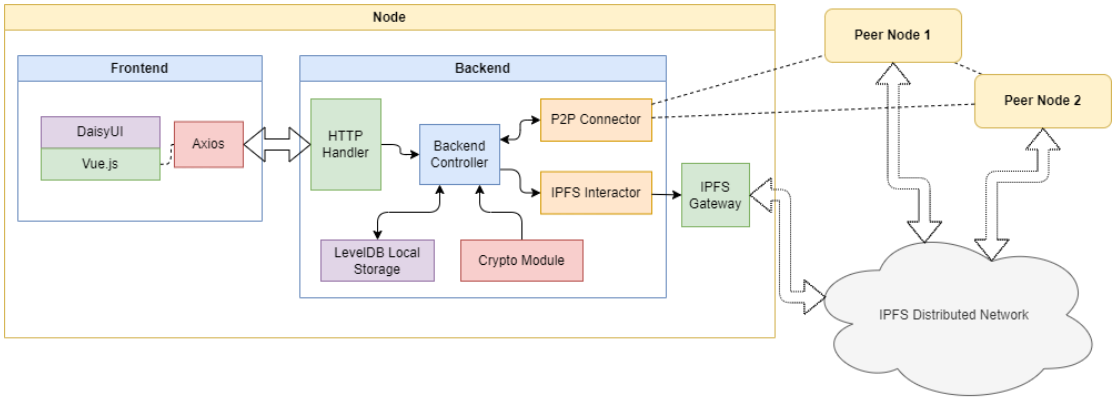


图 1

HTTP 模块：接收前端通信请求并分发，使用 Bearer Token 进行鉴权以防止攻击。后端所有

可用 API 如图 2 所示。

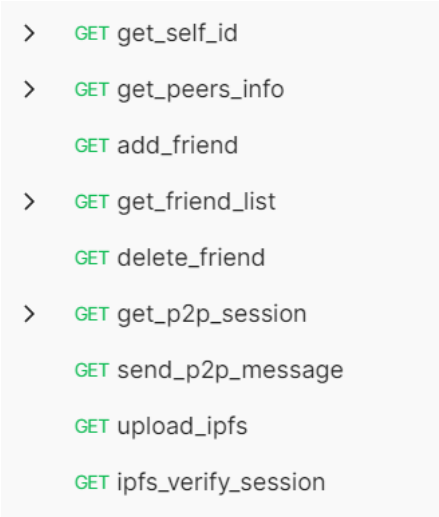


图 2

P2P 连接模块：负责发现网络上的对等节点，验证活跃状态并反馈给前端，向对等节点发送或接收加密的消息。

IPFS 交互模块：负责连接 IPFS Gateway 读取并验证消息。通过用户手动设定 IPFS API 进行上传，再由后端与 IPFS Gateway 交互，实现加密消息的分布式存储和索引。

LevelDB 本地存储模块：负责对发送和接收的消息进行本地离线缓存，用户在启动软件时无需等待后端连接 IPFS 网络，而是先从本地存储获取消息并显示。

加密模块：

在加解密过程中，首先获取发送方私钥和接收方公钥，接着，使用 ECDH 协议从私钥和公钥派生出共享秘钥，共享密钥在 ECC 里是一对共享的点，派生出共享密钥后进行 sha256 哈希作为加密密钥，然后采用 SM4 算法 CTR 模式进行对称加解密操作。加密过程添加了随机 IV 以保证安全性，采用随机 IV 的原因是块加密本身的限制，块加密使用相同的 IV 会产生相同的密文，可能会被重放攻击，或者被猜测原文内容。加解密流程如图 3 所示。

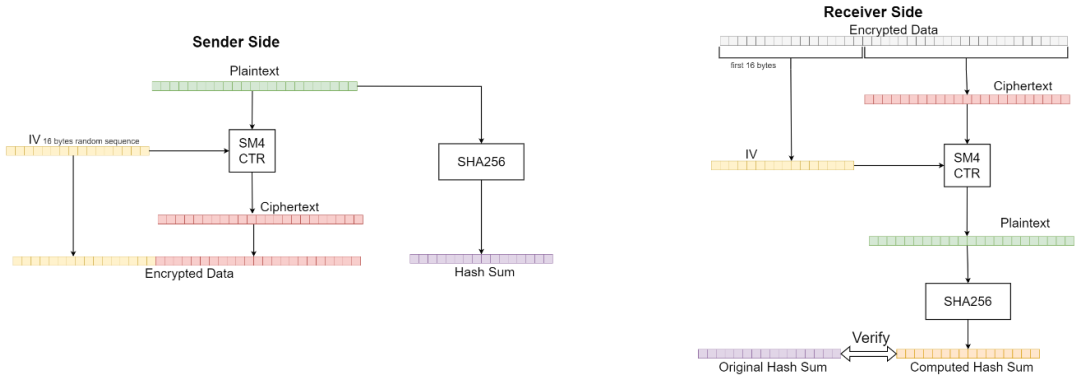


图 3

加密时，输入明文 plaintext 并生成一段 randomly IV，将生成的 randomly IV 与输入的 Plaintext 进行 SM4 CTR 算法加密，生成 Ciphertext，将 randomly IV 与 Ciphertext 合并为新的密文 Encrypted Data。解密时，将 randomly IV 与 Ciphertext 分离，然后经过 SM4 CTR 算法解密，即还原明文 plaintext。

三、 方案实现

环境：

- 系统：Linux Ubuntu 22.04
- Go 版本：1.20.5
- Node 版本：18.7.0
- Python 版本：3.10.7

基本运行环境：

进入 dapp-chat 项目根目录，运行位于 scripts 目录下的测试 P2P 后端启动脚本：

python3 ./scripts/run_test_p2p_network.py

初次使用会自动下载一些依赖包：

```
go: downloading github.com/ethereum/go-ethereum v1.11.5
go: downloading github.com/naoina/toml v0.1.2-0.20170918210437-9fafd6967416
go: downloading github.com/urfave/cli/v2 v2.17.2-0.20221006022127-8f469abc00aa
go: downloading gopkg.in/ini.v1 v1.67.0
go: downloading github.com/ipfs/go-ipfs-api v0.6.0
go: downloading github.com/rs/cors v1.7.0
go: downloading github.com/syndtr/goleveldb v1.0.1-0.20210819022825-2ae1ddf74ef7
go: downloading github.com/naoina/go-stringutil v0.1.0
go: downloading github.com/emmansun/gmsm v0.17.3
```

成功运行一个引导节点和两个常规节点：

```
#1 102e0de7d9586b40990d986e3c5baee68678a16b2d90af3a086fb8f048594541
#2 a7c32ffba6e9449229886522f8e37d5684fc95e07ad92d05dd3577922dcd0321
#3 436e3172a545d485587c0e75b18b1e0759d9154396a15511a34d23200c4d2c89
retcode 0
```

进入 frontend 目录，这个目录包含了前端的代码。使用 npm i 安装的模块和依赖到全局，接下来启动 vue 项目的前端服务。

运行 npm run dev 搭建脚手架环境。这个命令会去项目的 package.json 文件里找 scripts 里找对应的 dev ，然后执行 dev 的命令。

成功在本地的 44499 端口开启了 network 服务。

```
VITE v4.3.9  ready in 585 ms

➜ Local:   http://localhost:44499/
➜ Network: use --host to expose
➜ press h to show help

daisyUI 3.0.3 https://daisyui.com

✔ [ 1 ] theme is enabled. You can add more themes or make your own theme:
  https://daisyui.com/docs/themes

❤ Support daisyUI: https://opencollective.com/daisyui
```

通过配置文件启动：

配置文件 config.ini 如下：

```
; project data directory.
data_dir = chatdata2

[http]
; http server listener address.
; Example: 127.0.0.1:8080
address = 0.0.0.0:44445
; http server authorized token.
token = 130

[backend]
; marshaled secp256k1 private key.
; Example: 8d81af4001549f00aa0ea9c79a92dc716fa6f069a46615278529a88ff0970a57
private_key =
94fa73a584836623b488b0db234f5dee3147d85f04e383fdb8a474109c0f68f2
; peer listener address
; Example: 127.0.0.1:44445
address = 0.0.0.0:0
; bootnodes for discovery, seperated by ','
; Example: enode://...@host:port,enode://...@host:port
bootnodes = enode://
075c4cdca8604e3e5a5a67122615714a426435a11a3874b16d66fac604b08cc0ebc55c9019e
fe8ca464bf7c9e5bfc25ee8a4736e24a26a4e8adaadce0d45a7a5@192.168.72.129:44400
```

生成后端文件，并且使用配置运行：

./p2pchat-bin --config config.ini

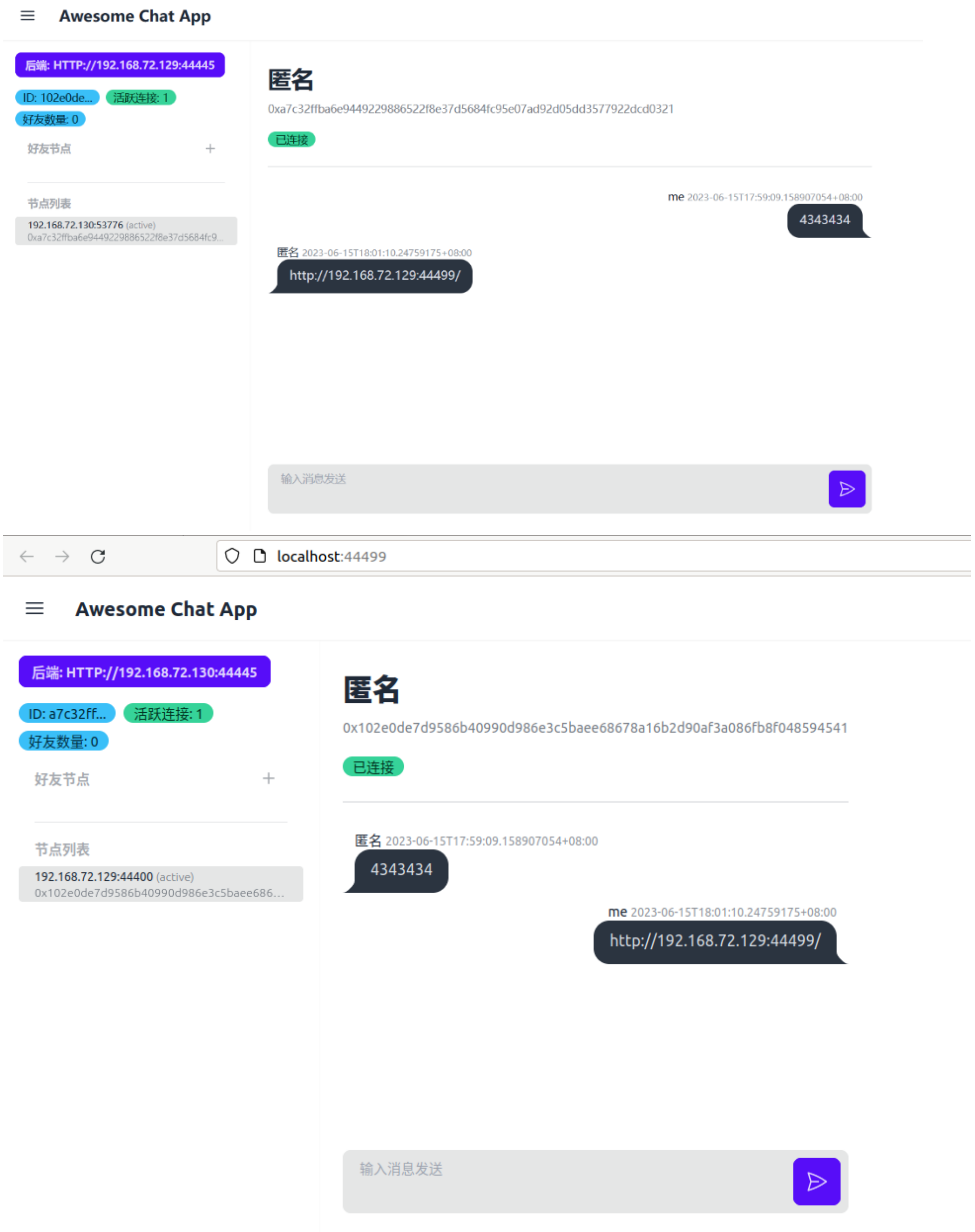
```
# go build -o p2pchat-bin ./p2pchat
# ./p2pchat-bin --config config2.ini
```

```
2023/06/15 17:58:19 leveldb initialized at chatdata2 with Node ID a7c32ffba6e9449229886522f8e37d5684fc95e07ad92d05dd3577922dcd0321
Geth DEBUG[06-15|17:58:19.465] UDP listener up                     addr=[::]:42251
Geth DEBUG[06-15|17:58:19.466] TCP listener up                     addr=[::]:42251
Geth INFO [06-15|17:58:19.467] New local node record               seq=1,686,823,099,454 id=a7c32ffba6e94492 ip=127.0.0.1 udp=42251 tcp=42251
2023/06/15 17:58:19 Started P2P networking at enode://63845e288672d58d5a30ac91783cda2c2be47dd5937d4b4d500fc90924f8908add1d6be87b6fea92b056ee6b71189f3d4cb9
86abead015c30d25b3e9c8d775ea@127.0.0.1:42251
2023/06/15 17:58:19 Node ID: a7c32ffba6e9449229886522f8e37d5684fc95e07ad92d05dd3577922dcd0321
2023/06/15 17:58:19 Started http server at http://[::]:44445 with token 130
Geth INFO [06-15|17:58:19.467] Started P2P networking              self=enode://63845e288672d58d5a30ac91783cda2c2be47dd5937d4b4d500fc90924f8908add1d6
be87b6fea92b056ee6b71189f3d4cb986abead015c30d25b3e9c8d775ea@127.0.0.1:42251
```

虚拟机通信测试：

通过上述配置文件，启动两个后端 http://192.168.72.130:44446 和 http://192.168.72.129:44445

配置完成后，测试发送消息：



IPFS 配置：

运行 IPFS：ipfs daemon

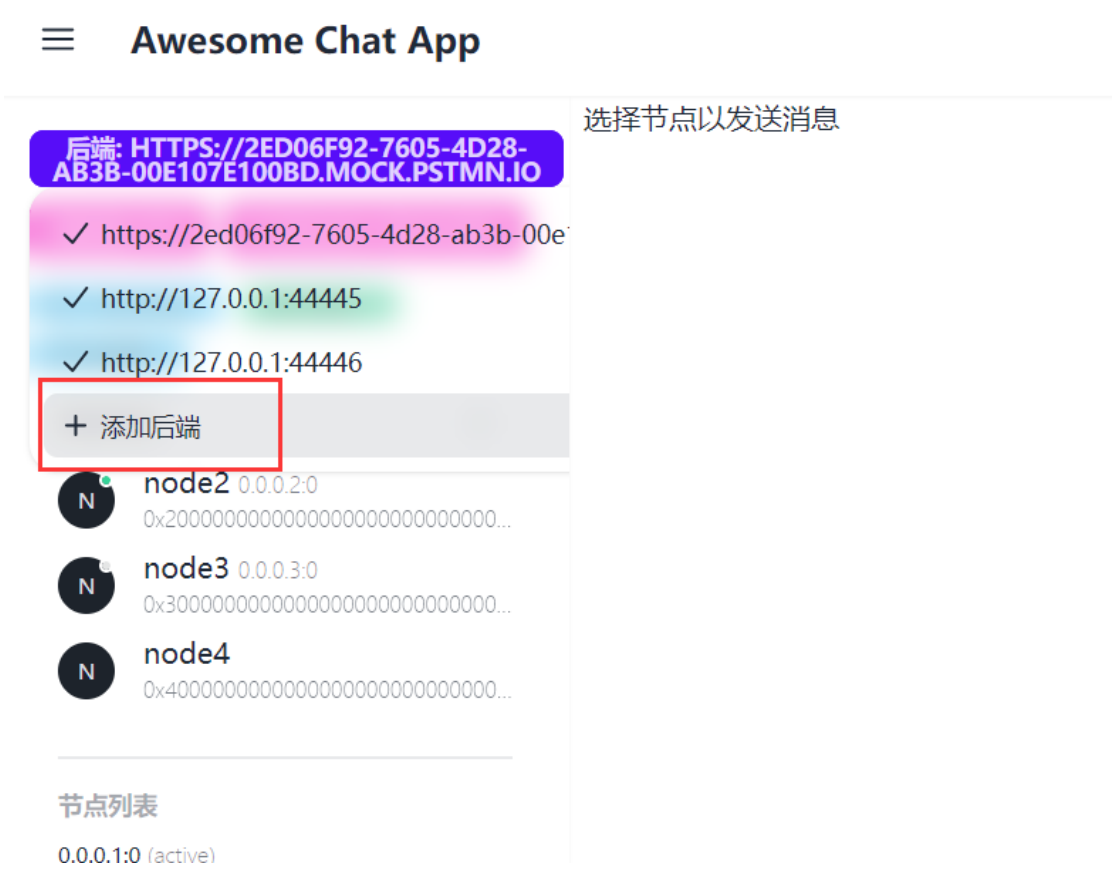
进入 http://127.0.0.1:5001/webui 访问 IPFS 的面板，找到配置文件，其中 API 地址是后面用户手动上传时用到的地址，Gateway 地址则是一个只读的文件代理，同样也要用于后面配置。

IPFS 上传和验证：

四、 运行结果/应用效果

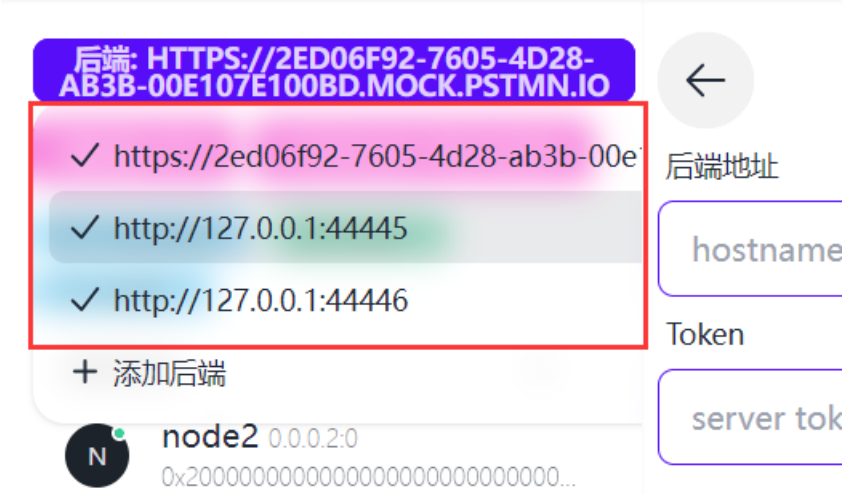
添加、切换后端节点

点击【后端：未选择】-【+ 添加后端】



输入后端地址以及 token，点击【添加后端】

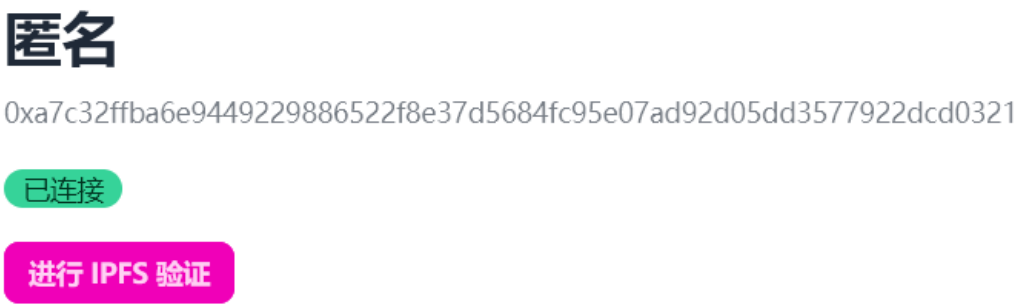
切换/选择后端节点，【后端：xxxxx】--【http://ip:port】



按钮介绍

同步至 IPFS（表单）：将当前后端记录同步至对应 IPFS 并发布至 IPNS

Storage Provider（表单）：设定 IPFS Gateway

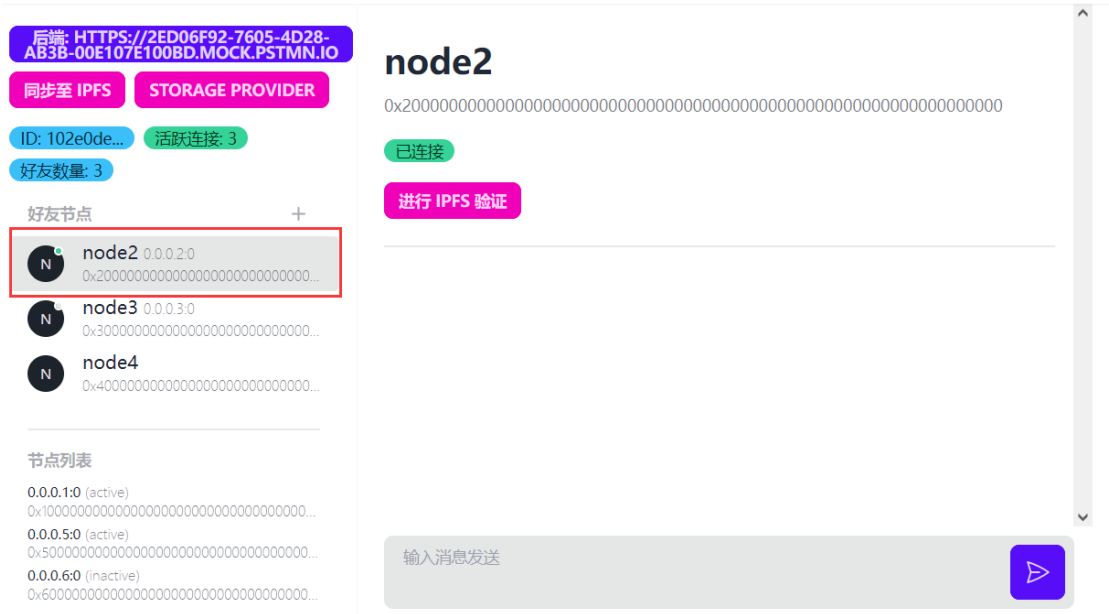


进行 IPFS 验证：单击验证当前会话的消息是否同步至 IPFS

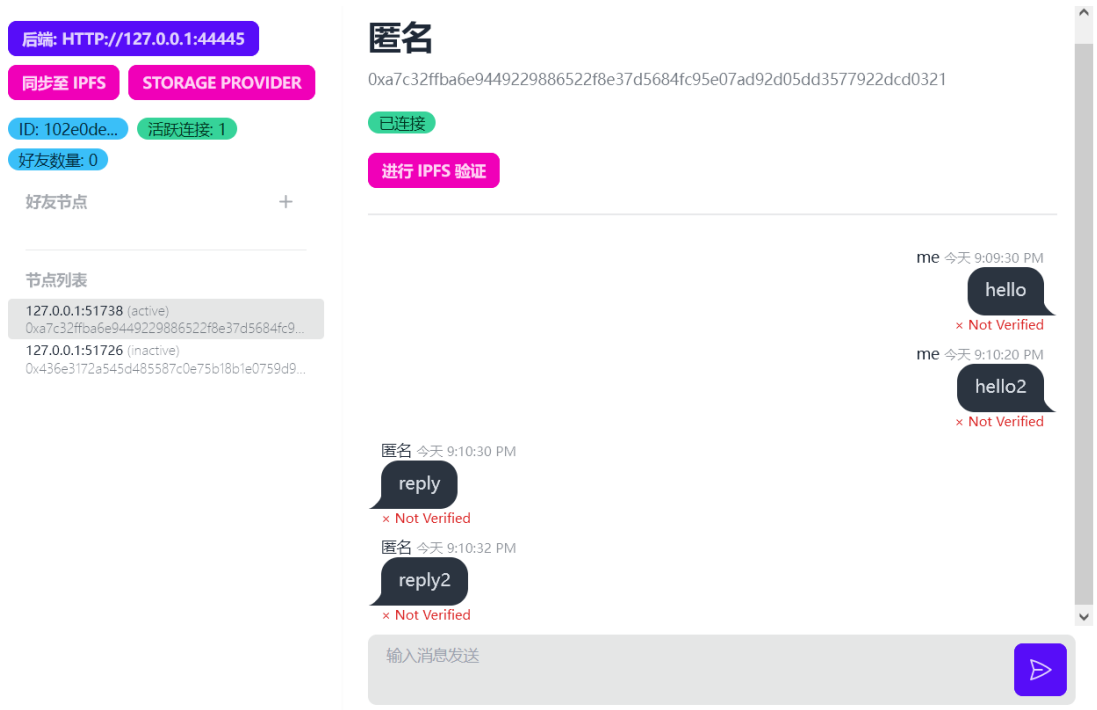
收发消息

待后端 P2P 网络有节点连接后，可以在左侧状态栏看到节点信息以及好友信息

在左侧选择一个好友或者匿名节点，进入聊天面板



发送消息，输入消息后按【enter】或者单击发送按钮

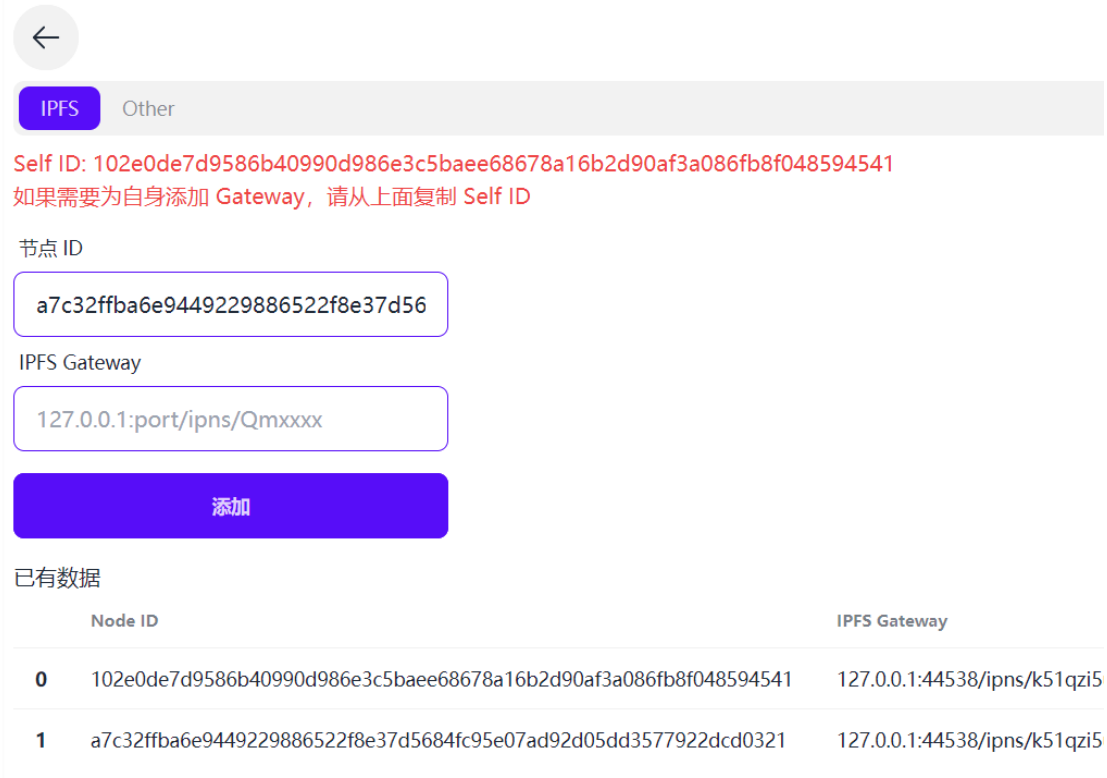


同步至 IPFS

填写好 IPFS API 地址后开始同步，显示【同步中】

同步完成后，会显示 IPNS 信息

设定 Storage Provider（IPFS Gateway）

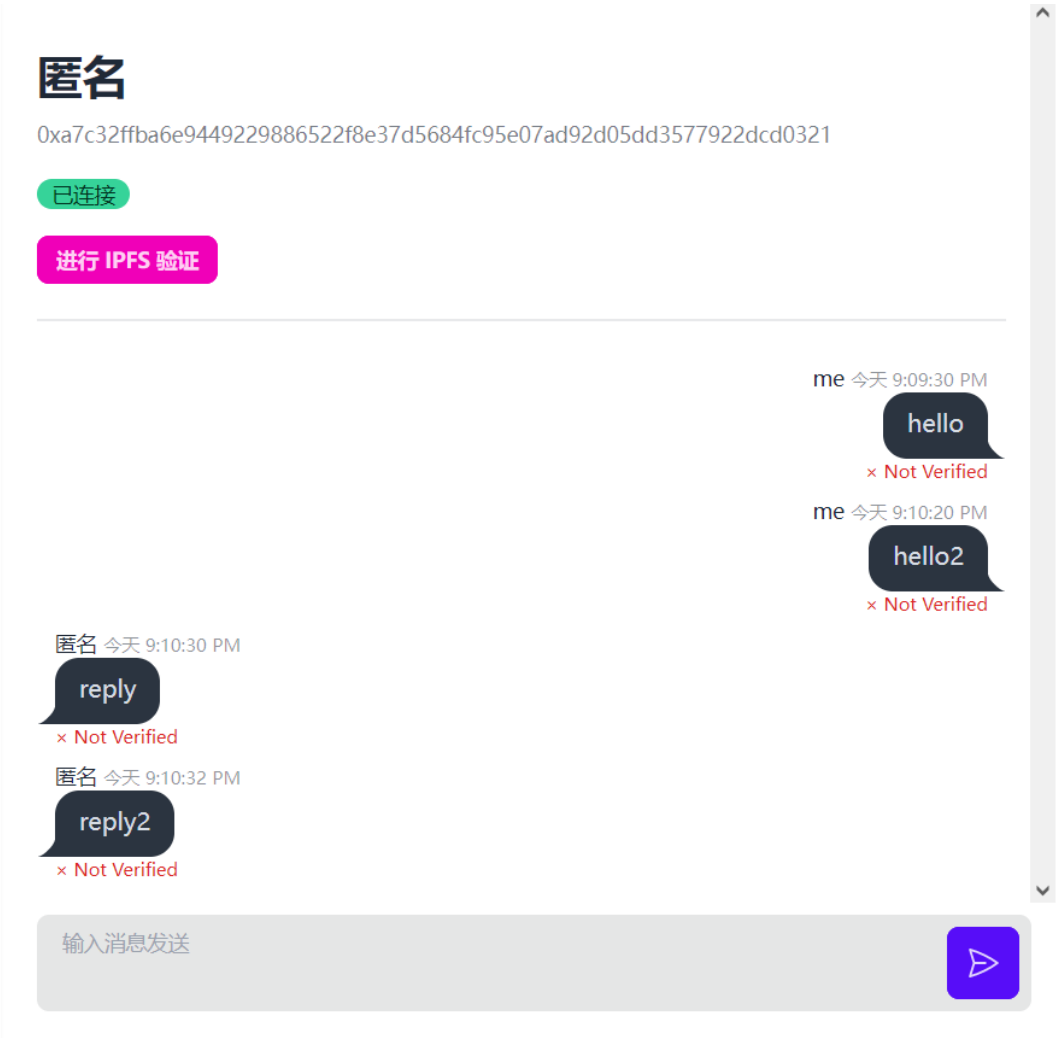


为节点绑定 IPFS Gateway，建议填写上一步【同步至 IPFS】拿到的 IPNS 地址。绑定完成后会在下面表格显示绑定信息

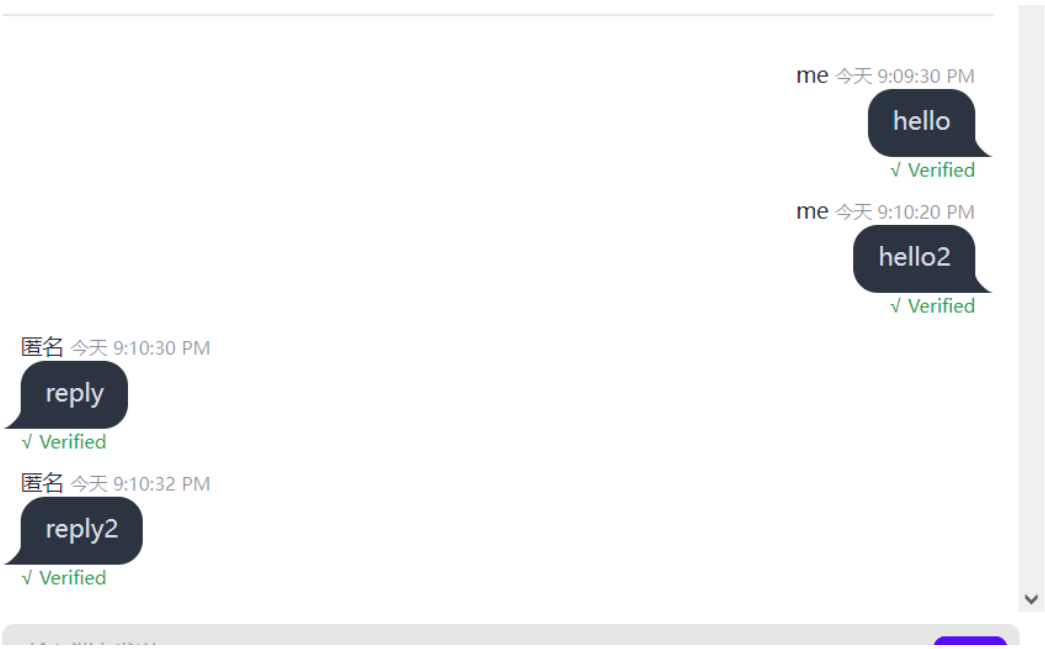
进行 IPFS 验证

进入一个聊天面板，并且使用两个身份互相发送测试信息。

确保已经完成前面的步骤【上传至 IPFS】和【设定 IPFS Gateway】



单击进行 IPFS 验证，会将后端返回的已验证的消息哈希和目前会话的消息哈希进行一一比对



五、 创新与特色

区块链协议支持的 P2P 通信模式：该软件采用区块链的 P2P 基础协议作为通信方式来实现消息传输和共享。每个客户端持有一个密钥对作为一个节点，发送消息会自动进行加密和签名验证，具有极高的安全性。

ECDH 交换密钥+ SM4 算法：ECDSA（Elliptic Curve Digital Signature Algorithm）交换密钥采用了椭圆曲线公私秘钥生成及交换方法。相较于传统 RSA 算法，在保证同等安全强度的情况下，其所需秘钥长度更短、并且采用对称加密获得的速度提升巨大。SM4 是一种国密对称密码学加解密算法。它采用分组密码方式来加/解密数据块，与其他流行的对称密码学算法如 AES 相比，SM4 拥有更好的性能表现和更高级别的安全性。

去中心化、持久化 IPFS 储存设计：利用 web3 中流行的 IPFS 分布式储存网络分散化储存聊天记录，设计了 IPFS 文件索引、交换密钥对称加密和哈希完整性，每个人只需要上传自己的记录到自己的节点上，并通过交换密钥加密，可以保证只有接收者可以解密对应记录。