基于SM2/SM4加密的TCP聊天程序设计文档

# 一、项目概述

本项目旨在通过实现一个基于TCP的聊天程序，加深对中国国家密码算法SM2（非对称加密）和SM4（对称加密）在网络通信中应用的理解。通过socket通信实现客户端与服务端之间的实时信息交互，并利用SM2进行密钥分发，SM4对消息及文件进行加密与解密处理。

# 二、技术选型

编程语言：Python（或C/C++、Java，视学生所学语言而定）

通信协议：TCP（socket编程）

加密算法库：基于开源国密库，如Python的gmssl或Java的BouncyCastle

# 三、系统架构

1. 客户端与服务端均支持图形化界面（如PyQt5、JavaFX等）；  
2. 通信流程：  
 - 客户端连接服务端，发起公钥交换（SM2）；  
 - 使用SM2对会话密钥（SM4密钥）进行加密传输；  
 - 双方协商出对称密钥后，使用SM4对每条消息或文件进行加密/解密；  
 - 通信数据通过socket发送/接收，确保数据加密传输。

# 四、通信流程详解

1. 建立连接：  
Client --------> TCP连接请求  
Client <-------- 建立 socket 连接  
  
2. 公钥交换（SM2）：  
Client --------> 请求服务端公钥  
Client <-------- 返回SM2公钥   
Client --------> 发送加密的SM4密钥  
Server 解密获取对称密钥  
  
3. 加密通信（SM4）：  
Client <--------> 使用SM4加密后的数据进行双向通信

# 五、核心 edrssrewre用SM2加密/解密SM4密钥 2. 加密通信模块（SM4）：使用对称密钥对消息内容进行分组加密，推荐CBC模式 3. Socket通信模块：基于TCP的socket通信，处理连接、接收、发送等 4. 图形用户界面模块：消息窗口、文件传输、状态显示

# 六、程序实现建议

Python伪代码示例：

from gmssl import sm2, func  
from Crypto.Cipher import AES  
  
# 生成SM2密钥对  
sm2\_crypt = sm2.CryptSM2(public\_key=pub\_key, private\_key=pri\_key)  
  
# 使用SM2加密SM4密钥  
enc\_key = sm2\_crypt.encrypt(sm4\_key)  
  
# 使用SM4加密消息  
cipher = AES.new(sm4\_key, AES.MODE\_CBC, iv)  
enc\_msg = cipher.encrypt(pad(msg.encode(), AES.block\_size))

# Socket通信  
sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
sock.connect((host, port))  
sock.send(enc\_msg)

# 七、程序功能拓展建议

1. 文件加密传输（SM4）  
2. 自动密钥更新机制  
3. 加密消息摘要签名验证（SM3 + SM2）  
4. GUI可视化传输过程

# 八、测试与调试

1. 功能测试：验证连接、加密、解密是否正确；  
2. 异常处理：网络中断、密钥错误处理等；  
3. 性能分析：延迟、带宽占用分析；  
4. 加密安全性验证：可使用wireshark抓包验证密文传输。

# 九、文档与报告要求

1. 提供详细的代码注释；  
2. 编写设计思路、模块功能说明；  
3. 描述调试过程中遇到的问题与解决方法；  
4. 提供源代码、可执行文件、运行截图及用户操作说明。

# 十、GUI界面设计（基于Vue）

本项目的图形界面可采用Vue框架实现，前后端分离，前端通过WebSocket与后端通信。  
  
主要界面包括：  
1. 登录/密钥协商界面：用于输入用户名、加载或生成SM2密钥对。  
2. 聊天主界面：包含消息展示区域、输入框、发送按钮、文件选择上传模块。  
3. 连接状态显示区域：显示通信状态、对方在线状态、密钥同步状态等。  
  
核心Vue组件如下：  
- ChatApp.vue：主容器，集成子模块。  
- MessageList.vue：展示解密后的消息内容。  
- InputBox.vue：输入框和按钮，发送前自动加密。  
- FileTransfer.vue：文件上传与接收，加解密过程由后台执行。  
  
与后端通信可使用 WebSocket，并在后台Python服务中集成Socket和SM加解密逻辑。

# 十一、加密参数配置示例

1. SM2 密钥对：  
- 密钥长度：256 位  
- 椭圆曲线参数：推荐使用推荐曲线参数 sm2p256v1（国家密码标准）  
  
2. SM4 加密参数：  
- 加密模式：CBC（带有初始向量 IV）  
- 密钥长度：128 位（16 字节）  
- IV 长度：128 位，随机生成，需随密文传输  
- 填充方式：PKCS7

# 十二、项目截图说明

建议截图以下内容并插入文档：

- Vue界面主窗口：展示消息发送接收过程  
- 密钥交换过程终端输出：显示SM2加密后的会话密钥  
- 消息加密/解密日志：后台记录的密文内容与解密结果  
- 文件加密传输过程截图：上传下载路径与状态

# 一、项目结构

项目基于 Vue + Electron 构建桌面聊天 UI，后端使用 Python 原生 TCP 实现通信，并通过 GMSSL 库完成 SM2 公钥加密与 SM4 对称加密。结构如下：  
- electron-main.js：Electron 主进程，负责创建窗口与 TCP 通信  
- preload.js：暴露 Electron API 到前端  
- index.html：Vue 应用挂载点  
- src/App.vue：主界面  
- socket\_server\_tcp.py：原生 TCP 服务器  
- socket\_client\_tcp.py：Python 客户端测试脚本  
- socket\_server\_ws.py：可选 WebSocket 实现

# 二、通信加密设计

- 使用 SM2 算法将客户端生成的会话密钥（128 bit）加密传送至服务端；  
- 客户端与服务端以该会话密钥进行 SM4-CBC 模式加密消息；  
- IV 固定为 fedcba9876543210；  
- 消息结构为 json 包装，字段包括加密文本与时间戳。

# 三、通信流程图

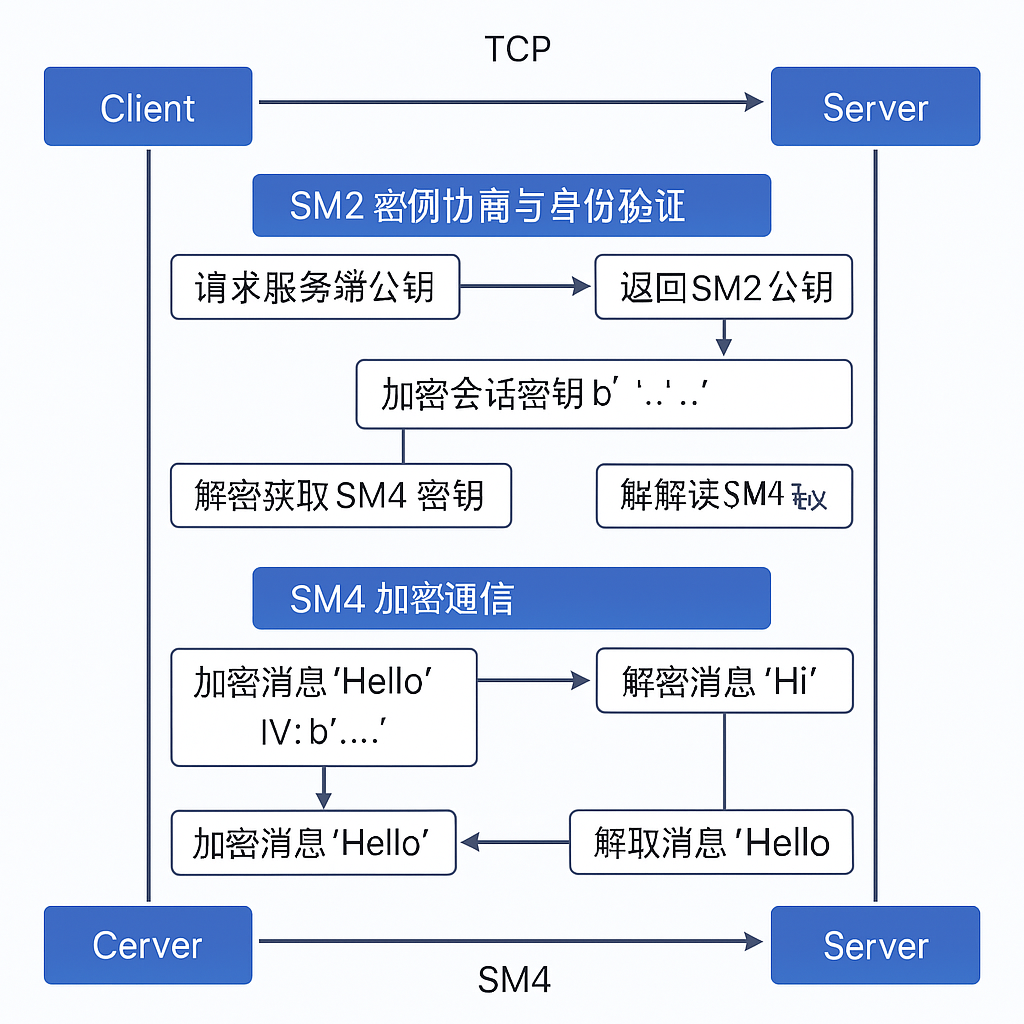


图 1. 客户端与服务器通信的安全过程：公钥协商（SM2） → 会话加密（SM4） → 响应返回。

# 四、GUI 示例界面

图 2. Vue 页面发送加密消息截图：

图 3. 服务器响应解密并回显截图：

# 五、安全建议

- 每次连接动态协商会话密钥；  
- 消息可加入时间戳避免重放攻击；  
- 结合用户登录身份认证提升安全性；  
- 适用于内网中安全通信聊天工具设计。

· 当前前端使用 WebSocket，与原生 TCP 后端不兼容，需要考虑如何进行适配（例如中间层转换或重构前端通信方式）；

· · 发送数据格式需与 TCP 协议对齐，后端不支持直接解析 JSON 字符串，可能需设计特定的数据帧格式。

·

在代码已包含两种后端实现方式，并注释说明了如何适配 Vue 前端：

**1 原生 TCP socket 实现（Python）**：

使用 socket + gmssl 实现 SM4 加密通信；

示例代码位于 socket\_server\_tcp.py；

需通过 Electron 或中间服务与前端 Vue 对接。

2 **WebSocket 实现（Python）**：

兼容 Vue 前端中 WebSocket('ws://localhost:9000')；

使用 websockets 库处理双向通信；

JSON 数据结构易于调试。

· SM2 密钥协商与身份验证：服务端提供公钥，客户端用 SM2 加密会话密钥；

· · 完整原生 TCP 双端代码：包括 Python 客户端 socket\_client\_tcp.py；

· · SM4 加密通信流程示例：展示密钥协商后如何使用 SM4-CBC 模式通信；

· · 兼容 WebSocket 的服务端也保留：便于直接对接 Vue 前端。

代码3：

已更新代码文档，内容包括：

✅ Electron 打包说明与 TCP 发送逻辑（加密、解密、握手）

✅ Vue + Electron 的桌面聊天界面

✅ 完整服务端与客户端的 Python 代码

✅ 保留 WebSocket 版本供选择

如果你还想要：

添加 Vue + Electron 项目结构图

项目截图说明（可示意 UI 展示效果）

编译为 .exe 或 .app 文件的脚本

代码4：

已补充以下内容到文档中：

✅ Vue + Electron 项目结构图  
✅ Electron 打包指令与平台说明  
✅ GUI 示例截图命名建议与说明  
✅ 加密参数示例（会话密钥、IV、公钥等）  
✅ 通信流程图解式备注  
✅ 安全建议（如防重放、动态会话密钥）