TLS project（思路）

- TLS协议
- socket
- DH协议
- 数字签名与验签
- 哈希算法
- Python 封装

V1（包含服务端，客户端A，客户端B）：

1. 服务端与客户端A B建立连接，进行客户端和服务端问候，数据传输
2. 身份验证：A,B端分别制作一组密钥，并将用问候消息使用hash算法然后用私钥加密后转换成数字签名，并和原消息一同传输给A(B)端（拥有公钥），B(A)用其公钥解密后的消息与原消息的hash做对比，一样则身份认证成功
3. DH协议协商密钥：服务端生成g，p发给A，B端，在A，B端分别生成各自私钥a，b，并用其计算公钥A，B，交换公钥，用各有的私钥再次进行模运算得到相同的密钥。作为AES加密密钥
   密钥确定后，打印，客户端就绪，服务器就绪
4. 消息加密与解密：A端发送消息前AES通过密钥进行AES加密，发送到B端，B端用密钥进行AES解密
5. 完整性
   计算传输消息的sha256，验证通信信息的完整性

V2（包含服务端A，客户端B）

A端：服务端

1. 配置日志通过 ：`logging` 模块将服务器活动记录到文件 `server_log.txt`
2. 生成服务端的私钥和公钥
3. 创建监听器
4. 接受客户端连接
5. 发送服务器公钥

创建服务端签名

- 对问候消息：生成哈希值
- 用私钥对消息哈希值加密

1. 发送服务端问候及其数字签名
2. 接受客户端公钥
3. 验证客户端签名

- 对问候消息生成哈希值
- 用公钥对签名进行解密
- 验证解密信息与原始哈希值是否相等
- 打印已确定客户端身份

DH密钥协商

1. 生成p，g
2. 发送DH参数
3. 用私钥计算公钥A
4. 发送公钥A
5. 接收公钥B
6. 用私钥和公钥B计算协商密钥

AES消息加密

使用协商好的密钥

调用`Crypto.Cipher`库的`AES`模块加密

验证消息完整性

B端：客户端

1. 配置日志：通过 `logging` 模块将服务器活动记录到文件 `client_log.txt`
2. 生成客户端的私钥和公钥
3. 连接服务器
4. 发送客户端公钥

创建客户端签名

- 对问候消息生成哈希值
- 用私钥对消息哈希值加密

1. 发送客户端问候及其数字签名
2. 接收服务器公钥
3. 验证服务器签名

- 对问候消息生成哈希值
- 用公钥对签名进行解密
- 验证解密信息与原始哈希值是否相等
- 打印已确定服务器身份

DH密钥协商

1. 接受DH参数
2. 用私钥计算公钥B
3. 发送公钥B
4. 接收公钥A
5. 用私钥和公钥A计算协商密钥

消息加密

使用协商好的密钥

调用`Crypto.Cipher`库的`AES`模块加密

验证消息完整性