基于安全通信协议的分布式数据爬取与多线程访问系统

关于通信安全与数据加密方法介绍

本项目设计了完整的数据通信与信息加密体系，涵盖三大安全模块：

- 通信加密（HTTPS）
- 身份认证（Token）
- 数据加密（RSA / AES / Fernet）

一．通信加密：HTTPS 协议

HTTP 发送的是明文 明文传输，易被窃听，为了保证安全性，

HTTPS 产生：在 HTTP 基础上用 TLS / SSL 进行加密 现在主要是 TLS

(注：HTTPS 不是 HTTP 的简单加密版，它是 HTTP 跑在 TLS 加密通道内的协议组合。)

先简单了解下类比生活“情书❤的传递”

小明暗恋小红，通过大胖传递情书给小红

问题：大胖可以在中间进行“偷看”，“篡改”

于是引入第一种方法——对称加密算法

首先什么是对称加密算法（这里通俗介绍，详细可见 AES 部分）

小明和小红之间有一套约定的加密规则 （eg: if English letter: then +1 letter）

小明 ： "I love you". 大胖：”M mpwf zpv” 大胖看不懂

大胖："M mpwf zpv". 小红：用与小明指定的规则解开了"I love you".

问题：如果小黑，小白……都喜欢小红，那么小红将存储多个规则（密钥）费空间

于是引入第二种方法——非对称加密算法 （RSA）

公钥私钥 （公钥公开，私钥绝不公开； 公钥加密，私钥解密； 公私钥成对出现）

小明先请求获取小红公钥 -> 小红发送公钥

小明用小红公钥进行情书加密 -> 小红用自己的私钥进行解密

这样就很好了吗？

问题：非对称加密算法效率太低太慢，差几百倍，而且随着加密内容的增加，非对称加密速度效率呈线性下降

于是引入第三种方法——混合加密算法 （非对称+对称）HTTPS

小明先请求获取小红公钥 -> 小红发送公钥

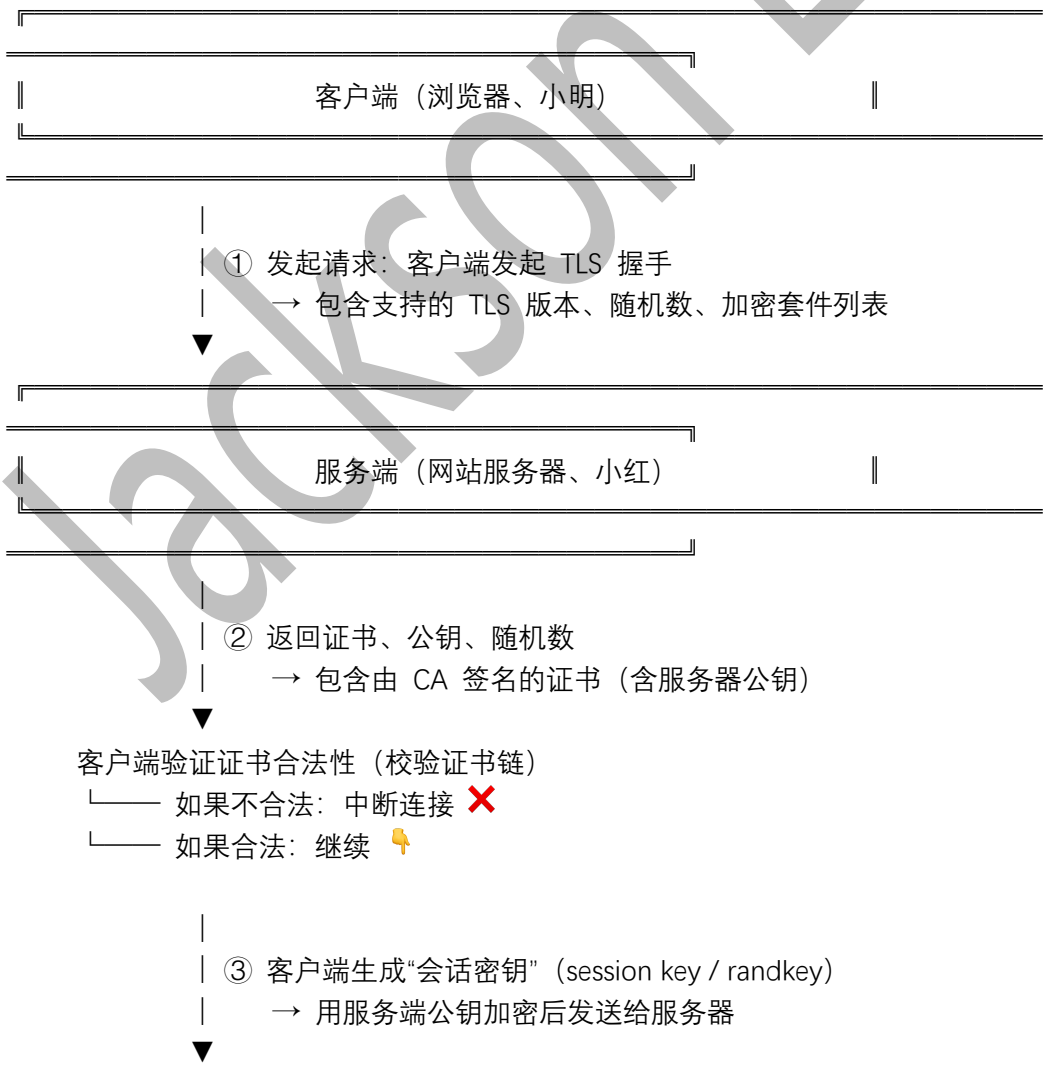
小明自己生成随机生成 Randkey 通过请求的公钥对 Randkey 进行加密 发送 randkey

小红拿到后用自己的私钥解密得到 Randkey ——非对称加密

小明用 randkey 对情书内容加密发送 -> 小红用 randkey 解密情书内容——对称加密
这样完美了吗?
问题：大胖可以自己做一套公钥私钥，在小明申请公钥时，把自己的公钥发给小明；同样套路他也可以伪造小明给小红发信息。
中间人身份伪造问题 + 中间人私发公钥问题

于是可以用第三机构来颁发证书 (SSL 证书)
CA 机构：Certificate Authority
小红向 CA 申请证书 小明拿到自己本身操作系统自带的根证书
小明申请向小红申请公钥 -> 小红返回给小明证书及公钥
小明拿自己的根证书检验小红证书确认发件人小红后完成对小红公钥存储
（后面就与混合加密算法一致了）
小明自己生成随机生成 Randkey 通过请求的公钥对 Randkey 进行加密 发送 randkey
小红拿到后用自己的私钥解密得到 Randkey ——非对称加密

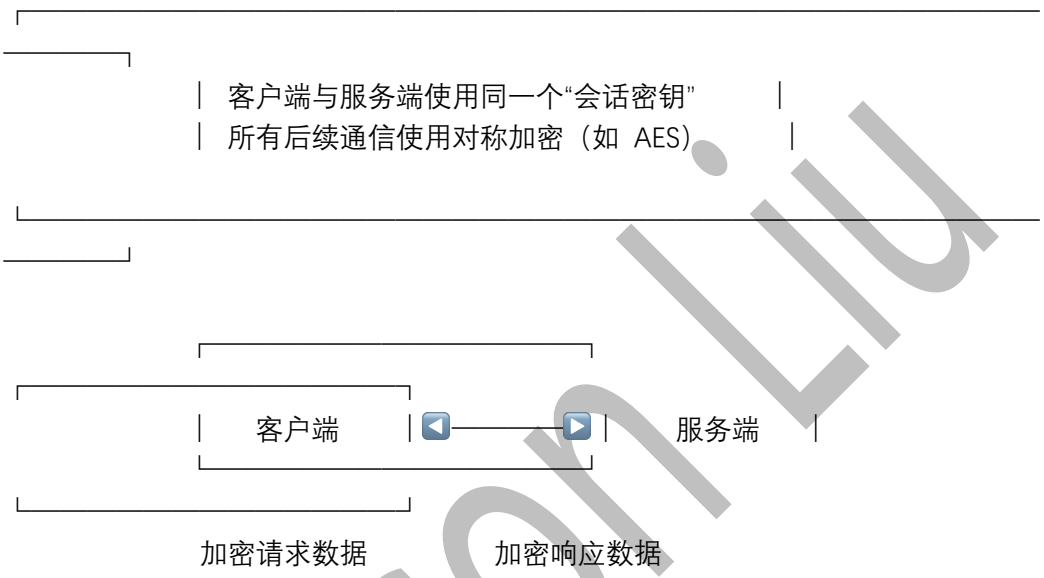
小明用 randkey 对情书内容加密发送 -> 小红用 randkey 解密情书内容——对称加密



服务端用私钥解密，得到“会话密钥”

④ 握手完成，建立安全通道（TLS session）

正式数据传输阶段（双向对称加密）



以上是第一部份，最终我们用 HTTPS + CA 证书可实现通信安全

- 注：本项目并没有实质性模拟 HTTPS + CA 证书 而选择用免费证书来提高性价比

二． 数据加密（RSA / AES / Fernet）

对数据本身进行加密 在本项目中我们就是对 token 登录密码那些进行数据加密

注：本文不详细讲述 Fernet 重点讲述 RSA & AES

对称加密算法——AES

这部分算法我觉得这个博主讲的太好了 完全没有超越空间了，直接看这个链接吧

AES 加密算法讲解视频——b 站

Fernet (AES 的高级封装，在数据库安全, Token 等领域常用)

非对称加密算法——RSA

基本数学概念：防止忘了

因数：a * b = c. a / b 是 c 的因数

质数：因数只有 1 和本身

余数不说了

RSA 非对称加密

公钥 (7,33)

源数据: C A O

十进制: 3 1 15

(suppose)

求幂: 3^7 1^7 15^7

(2187, 1, 1708593375)

求余: (2187%33, 1%33, 15%33)

(9, 1, 27) 密文发送.

私钥 ($\hat{3}$,33)

密文: 9, 1, 27.

求幂: 9^3, 1^3, 27^3

求余 $9^3\%33$, $1^3\%33$, $27^3\%33$

3, 1, 15

C A O

(7, 33)
E, N

(3, 33)
D, N

明文^E%N=密文.

密文^D%N=明文.

Part 2. 公钥. 私钥制作过程.

① Pick 2 质数. p=3, q=11

② 质数相乘: N=p×q=33 公私钥相同数字

③ 欧拉函数: T=(p-1)×(q-1)=2×10=20 important.

④ 选公钥E: 3条件: 1)质数; 2) 1<公钥<T; 3)不是T因子.

⑤ 算私钥 D: (D×E)%T=1 E=3公开 → 公钥(3,33) D=(7,33)

三． 身份认证（Token）这里用 Readme 写了一个

🔐 Token 身份认证机制

本项目使用基于 Token 的身份认证方案，以确保 API 接口在分布式多线程访问时的访问安全、权限控制与身份识别。

🧭 1. Token 机制简介

Token 是由服务器颁发的**加密身份凭证**，客户端登录成功后获取并在后续请求中使用。

Token 的作用本质上是：

> 用加密算法生成的“电子身份证”，通过 HTTPS 通道传输，API 通过它判断请求是否合法。

💡 2. Token 与 API 的关系

元素	说明
API	提供服务的功能入口，如 `/api/user/info`
Token	判断访问者身份和权限的通行证

每次访问 API 时，必须在请求头中携带 Token：

```
GET /api/user/profile
Authorization: Bearer <token>
```

API 会通过解密 / 验签方式验证 Token 并决定是否放行。

✅ 总结一句话：

> **Token 是钥匙，API 是门。你要访问 API，就必须拿对钥匙。**

🔁 3. Token 工作原理（模拟故事版）

> 人物设定：
>
> - 小明：客户端用户
> - 小红：服务器/API 服务端
> - 大胖：中间人（可能偷听）

① 登录获取 Token

小明发送用户名和密码给小红：

```
POST /api/login
```

小红验证成功，生成一个加密后的 Token（如 JWT 或 Fernet），发还给小明。

② 后续访问 API 时带上 Token

小明访问数据接口时：

```
GET /api/data
Authorization: Bearer eyJhbGciOi...
```

小红解密 Token，确认身份与权限后，决定是否返回数据。

🔐 4. Token 是什么的“综合体”？

功能层	Token 的角色
🔒 加密层	Token 内容由加密算法生成（如 Fernet、JWT）
🪶 通信层	Token 通过 HTTPS 加密通道安全传输
🚪 API 控制层	Token 是服务器判断请求者身份和权限的核心依据

🧩 5. Token 安全建议

- 所有 Token 通信必须通过 **HTTPS**，禁止明文 HTTP。
- 设置 Token 有效期，避免长期有效（如过期时间 + Refresh Token）。
- Token 生成应采用强加密签名，如：
  - HMAC-SHA256（JWT）
  - AES-HMAC（Fernet）

🧪 6. 完整流程回顾

```
① 登录：
    客户端发送账号密码 → 服务端验证成功 → 返回加密 Token

② 使用：
    客户端每次请求 API → 请求头中附带 Token

③ 验证：
    服务端解密 Token / 验证签名 → 判断是否合法 → 决定是否返回数据
```