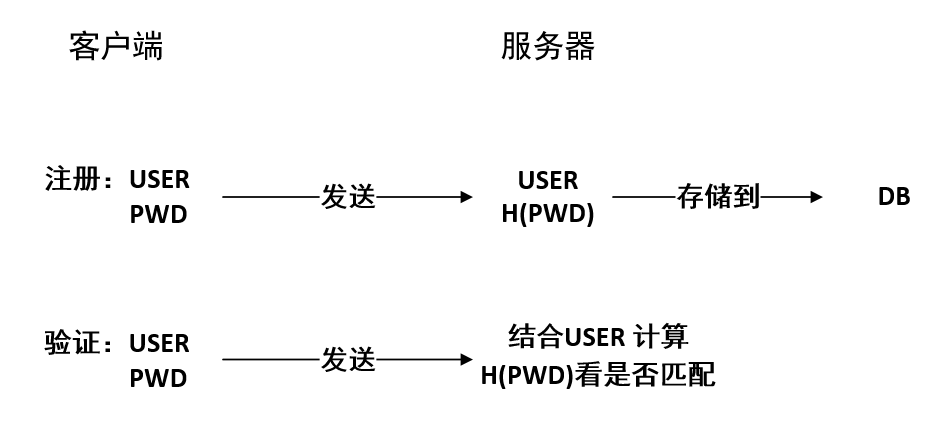
**需要增加的功能**

1. **增加用户验证功能：**

方案：



用户注册时输入用户名和密码之后，发送到服务器，服务器做接收数据，并求密码的消息摘要，然后将用户名和密码的消息摘要存储到数据库中；

// method post data:{ USER: string, PWD: string}

// return:{ status: 'success（插入成功）|failed（插入失败，可能是用户名已被占用）'}

URL：UrLPrefix +‘Register’

前端发送数据传JSON，登录验证时，服务器结合USER用户名和密码的消息摘要进行查询匹配，如果匹配，返回TRUE，则登录成功，否则返回FALSE登录失败；

// method post data:{ USER: string, PWD: string}

// return:{ status: ‘success’（登录成功）|‘failed’（登录失败，可能是密码错误）'}

URL：UrLPrefix +‘Login’

1. **增加展示辅助数据库**

数据库名：**DemoAuxDB**

数据表有三个：

1. **UserInfo**

这个表存储用户信息，包括两个字段：



分别是用户名和口令信息;

1. **ExpRecordLog**

这个是数据操作时间相关的数据表，字段如下：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| DateTime | Algorithm  (算法) | OpType（操作类型） | OpName（操作名称） | Execution\_time | Communication\_time |

通信时间=总时间-操作时间；

//method: post data: {Algorithm:‘SEMI’|‘FULL’， OpType: 'ADDITION' | 'MULTIPLICATION'| 'AND' | 'OR'| 'NOT' | 'XOR' | 'NAND' | 'COMPARE'|'SCALARMULTI'| 'SORT'| 'AVG'| 'STD'| 'LINEARREGRESSION', OpName: string，Execution\_time: float, Communication\_time: float}

//return { status: ‘success’}

URL：UrLPrefix +‘InsertExecuteTime’

（2）时间查询的数据接口：

前端请求，发送查询数据的时间区段，服务器返回对应时间区段的操作时间值

比如，服务器在某个时间区段内对于某操作的操作时间的查询：

select \* from table1 where DateTime between DateTime1 and DateTime2 and OpType = ‘ADD’ ；

允许查询所有ADD操作的操作时间：

select \* from table1 where OpType = ‘ADD’;

允许查询在两个操作时间点之间的所有操作的操作时间：

Select \* from table1 where DateTime between DateTime1 and DateTime2；

允许查询所有时间内所有操作的操作时间：

Select \* from table1；

// method: post data: {DateTime1:timestamps, DateTime2:timestamps, condition: string}

注：（1）如果是查询所有时间内的操作时间数据，那么只需要DateTime1或DateTime2为空;

（2）condition输入“OpType=‘ string’,Opname=‘string’，Algorithm=‘string’”；

//return {result: array( )}和数据库查询明文返回格式相同；

URL：UrLPrefix +‘QueryExecuteTime’

1. **EncData**

这个表示存储同态计算的数据密文的表，以便进行搜索查询，字段如下：



数据插入的操作在进行二元运算时进行，前端把两个密文传回给服务器，在做运算之前把两个密文直接插入到数据库中；

注：在同态计算时，前端增加对于当前登录用户的字段user:string;

1. 查询支持平均值、均方差、排序和线性回归的计算查询；

增加的URL（所得结果解密时调用之前的解密接口即可）：

//method post data:

{field:Xdata|Ydata,DateTime1:timestamps,DateTime2:timestamps,condition:string}

//return { data : array(number) ，result : string}密文

URL：UrLPrefix +‘QueryAveData’#

//method post data:

{field:Xdata|Ydata, DateTime1:timestamps,DateTime2:timestamps,condition:string}

//return { data : array(number) ，result : string}密文

URL：UrLPrefix +‘QueryStdData’

//method post data:

{field:Xdata|Ydata, DateTime1:timestamps,DateTime2:timestamps,condition:string}

//return { data : array(number) ，result : array(number)}

URL：UrLPrefix +‘QuerySortData’

//method post data: {field:Xdata，field2:Ydata，DateTime1:timestamps,DateTime2:timestamps,condition：string}

//return { data: array({x : number, y : number}) ，intercept: number, slope: number}密文

URL：UrLPrefix +‘QueryLinearRegressionData’

注：如果DateTime1或DateTime2是空，那么就是查询所有时间内的值；

计算查询的交互流程：

1. 前端发送查询请求，数据包括：数据表项（比如Xdata）和条件 condition（比如：where ID between 1 and 5 ）
2. 服务器进行数据库查询操作，得到符合条件的密文列；
3. 服务器做操作，对查询的密文列进行字符串拼接，然后使用函数进行OpType指明的操作，得到密文的计算结果（比如:求平均值的密文结果）；
4. 将查询得到的密文列以及计算得到的密文结果返回给前端，前端做展示；
5. 前端发送请求解密计算得到的密文结果，服务器将解密的结果返回给前端显示；

问题：

* 1. 全同态加密的接口设计；