# 隐语开放标准

开放标准：一系列协议的集合

包括数据、组件、节点执行、运行报告等协议

参考：secretflow/spec

[欢迎访问隐语开放标准 | 开放标准 main | 隐语 SecretFlow](https://www.secretflow.org.cn/zh-CN/docs/spec/main/)

## 数据

DistData = Public Data + DataRef

所有数据都是DistData（分布式数据），有2部分：

Public Data: name, type, meta, system\_info

DataRef: ownership + uri (相当于指向私有数据的指针)

DistData是public的。一些案例：

MPC的replicate secret sharing中，数据的分片都是DataRef

FL中联合表的内容也是DataRef指向的remote object

完整调用流程：

DataRef的owner根据uri和private storage config，得到真实存在的remote object

## 组件

组件的组成：

domain: namespace

name: unique in the namespace

version

(这三个可以唯一确定一个组件)

attributes: 属性

inputs

outputs

组件查询：[SecretFlow 组件列表 | SecretFlow v1.6.1b0 | 隐语 SecretFlow](https://www.secretflow.org.cn/zh-CN/docs/secretflow/v1.6.1b0/component/comp_list)

或者secretpad功能、comp\_list.json文件也可以查看

### 属性

(A) Atomic Attribute: 具体问题，如psi协议、分桶树、是否broadcast

(S) Struct Attribute: 一个模块，如ECDH，里面会有ECDH所有需要设置的信息

(U) Union Attribute: 分叉，不同协议需要的配置不同

一般的结构：S(PSI) -> U(ECDH) -> S(ECDH\_curve) -> A(curve\_25519)

### IO

需要指定类型（可以有多种）

如果是表格，可以指定列

可以对某一列填入指定的参数

### 节点执行

节点：组件的实际案例

节点内容：

组件定义

Storage Config (private for each participant)

NodeEvalParam:

组件定义中attribute的实例

Input: DistData

Output: 指定ID，生成到什么地方去

* NodeEvalResult

### 运行报告Report

数据结构：（和前端框架比较相关）

Descriptions: 以组的形式显示多个字段

Table: 显示数据行

Div

（页面的一个部分/节，由descriptions, tables或者divs组成）

Tab: 由div组成

Report：由tabs组成

# 调用组件

## Python API

SecretFlow Lib/Cli

Cli需要把命令通过base64编码，一般用于内部测试

一个组件调用的实际案例，参考psi\_demo\_secretnote\_150b0.pyu

## Kuscia

利用kuscia api

先把数据复制到容器内

注册数据的相关信息

对数据进行授权（如果授权成功，可以从一方节点查到另一方的数据）

执行隐私求交调用

使用kuscia api或者kubectl查看任务状态，成功后可以看到数据

## SecretPad

懒得用图形化界面(●ˇ∀ˇ●)

# 新增组件

参考网页：[隐语SecretPad平台新增组件 | docs v1.6.1b0 | 隐语 SecretFlow](https://www.secretflow.org.cn/zh-CN/docs/secretpad-all-in-one/v1.6.1b0/more_tutorials/new_components)

1. 在/secretflow/secretflow/component/下面新建compare.py，写入ss\_compare组件
2. 在/secretflow/secretflow/component/entry.py里面新增ss\_compare
3. 执行env PYTHONPATH=$PYTHONPATH:$PWD/.. python update\_meta.py更新组件列表
4. 打包镜像：bash build.sh -v cmp
5. 在Kuscia中注册自定义算法镜像