# SPU实现的PSI介绍

## PSI定义和分类

PSI: 安全求交集/隐私求交

是一种特殊的MPC协议

协议可以获取双方集合的交集，不泄露交集以外的信息

理论基础：MOOC一期

安全求交集和匿踪查询，BV1oW4y1H7rL

分类：

1. 根据参与方数量：两方/多方
2. 根据参与方数据量差异：balanced/unbalanced
3. 根据参与方恶意程度：半诚实/恶意
4. 带计算的psi类型：psi交集数量、payload分析、circuit psi

## sf PSI功能分层

分为以下三层：

1. PSI功能封装

MVP(最小功能实现)或者Kuscia: 白屏应用、openapi

SCQL: sql join(调用SPU C++接口)

Secretflow: psi\_csv, psi\_df

1. PSI协议实现(SPU):

bucket\_psi入口(屏蔽不同协议的差异，指定协议和输入数据就可以调用)

ECDH、KKRT16、BC22等协议

1. 基础组件(YACL):

OT, OPRF, PRG

ECC, AES, HASH

Brpc link

## SPU实现的PSI

PSI发展趋势：

基于公钥: ecdh, ec-oprf

=>基于OTE和布谷鸟哈希: KKRT16

=>基于Pseudorandom Correlation Generator: BC22, RR22 Blazing Fast

目前主要实现的是半诚实协议。

两方：

ecdh, kkrt16, bc22(pcg-psi)

ec-oprf PSI(unbalanced)

dp-psi

多方：

ecdh-3-party

恶意模型：mini-PSI（适合小数据集）

### ECDH

协议简单，易于理解

可以扩展到求交集数量、计算PSI

ECDH的特点：

通信量小，计算量大

SF的实现有三个特点：

1、性能提升：

支持了25519、FourQ曲线

增加了Intel AVX512处理器的multi-buffer支持

2、测评和合规要求：

增加了SM2曲线和256K曲线的支持

3、互联互通协议：

完成了跨平台互联互通开放协议中ECDH-PSI协议相关的标准

同样的环境下（LAN，2^24数据集）

25519速度比fourq快一倍

ecdh的fourq和kkrt协议差不多快

### KKRT

主要贡献：扩展了OT，并且构造了batch related oprf(BaRK-OPRF)

使用了cuckoo hash

有点：运行时间快，是最新PSI论文中比较的基准

缺点：内存占用较大、通信量大

SPU KKRT性能优化：

1. 矩阵转置 -> Bit Matrix Transpose (依靠intel CPU的指令实现加速)

还有Eklundh矩阵转置

1. AES -> Pipeline/Vector AES
2. CuckooHash -> Stash-less (stash计算量偏大)

### BC22

主要技术：

基于sVOLE的BaRK-OPRF

Generalized Cuckoo Hash(GCH)

PermutationBased Hashing

GCH把一行的元素增加到2-3个，在哈希数量小（如2）的情况下容易做到stash free

执行流程：

1. 基于VOLE的方案构建若干VOLE，具体数量根据cuckooHash数量、每行元素数量计算
2. 插入cuckoo hash和simple hash
3. 构建bark oprf，交互oprf的值，计算出交集

实现用到的参数：

GCH(3,2)每一行3个元素、2个哈希数量

VOLE使用emp-zk中的WYKW21 Wolverine方案

优点：通信量、运行时间均好于ecdh、kkrt16

（RR22性能比BC22更好）

### Unbalanced: ec-oprf based

适用场合：双方数据量差异大，例如一方billion, 另一方million

特定场合下，比普通的ecdh性能提升100%，原因是大数据集的计算量少了一半

基于一种椭圆曲线的oprf构造，可以减少计算量

### Unbalanced: SHE based

基于label psi的方案

封装了微软github上的相关实现

流程：

1. 服务器会计算差值多项式
2. 客户端把要查询的同态密文发送给服务器
3. 服务器计算多项式的结果，返还给客户端。
4. 客户端解密出来，如果是0，则说明服务端有相关数据。如果是一个随机数则表明服务器没有相关数据。

和ec-oprf比较，

优点是不需要把大数据方的数据传输过去。

缺点是计算量比较大

### 基于ECDH的三方协议

会泄露交集数量

AB先执行普通的ECDH求交，A把交集发送给C做进一步计算

### PSI相关文件

这部分的内容可能有变化。PSI已经有独立的代码库了

实现位置:

libspu/psi/core/\*

外部依赖：

yacl/bazel/repositories.bzl

yacl/bazel/libsodium.BUILD

bazel/emp\_tools.BUILD

bazel/Microsoft\_apis.BUILD

单元测试：

libspu/psi/core/\*test.cc

# SPU PSI调度架构

secretflow psi\_csv api

=>SPU: 分桶调度bucketPSI封装 => 具体PSI协议

分桶调度：千万求交->百万求交，结束后合并

bucket\_psi: 高级API，支持分桶，覆盖生产全部覆盖生产级全流程

（数据查重、分桶求交、结果广播、结果排序）

mem\_psi: 低级api，算法内核级性能+统一接口

operator: 算法接入层

bucket默认分桶是100万

关于batch\_provider

IBatchProvider是批量读取csv文件的基类

通过CsvBatchProvider可以读取csv文件

通过MemoryBatchProvider可以内存数据

# SecretFlow PSI开发

目前的python接口主要通过secretflow进行调用

secretflow有集群仿真模式，还有生产模式。

在集群仿真模式下，每一个ray节点模拟一个机构

生产模式下，每一个参与方需要同时执行代码才能完成任务。

生产模式下启动

1、需要先启动双方的ray集群：

ray start --head

2、然后初始化secretflow cluster config（地址、监听端口）

3、初始化spu设备：link和端口

4、配置spu.psi\_csv()的参数

# SF PSI后续计划

Psi协议开发：

Blazing fast协议

Circuit psi

Multiparty psi

Malicious psi

Psi调用框架。

独立代码库

优化入口函数和参数

优化协议封装架构

Psi产品化。

轻量化部署

算法原理可视化