PIR -> private information retrieval -> 隐匿查询

# SF PIR总体介绍

## 定义和分类

定义：

用户查询服务器数据，服务器不知道查到了什么

分类：

1. 按服务器数量：

单服务器方案

多服务器方案

1. 按查询类型：

Index PIR

Keyword PIR

SF目前支持：仅限单服务器

Keyword pir: labeled psi

Index pir: sealPIR

## 实现位置

（目前已经和psi单独封装到独立的代码库了）

1. 功能封装：(secretflow)

pir\_setup

pir\_query / pir\_mem\_query

1. 协议实现层：(SPU):

Label psi实现

sealPIR实现

1. 基础组件层：(YACL):

OT, OPRF, PRG

ECC, AES, HASH

Brpc link

## 操作流程

1. pir\_setup()

是单方任务，服务器单方面加密

1. pir\_query()

双方执行查询任务

# Index PIR – SealPIR介绍

## 基本原理

目前sf实现基于COPR（基于BFV的动态方案）

相关参数：多项式次数N、明文模t（一个N维向量）、密文模q

主要运算：密文加法、明文\*密文、替换

乘法的噪声增长比较大，COPR已经尽量少用了乘法

He实现index pir的基本原理：（问题是n比较大）

客户端的查询向量（n个密文向量）用he加密发给服务器

服务器计算内积，得到待查询数据的加密结果

客户端解密得到数据

## SealPIR贡献

1. 多个数据pack到一个同态明文

**案例：原始数据B1-B3打包到一个多项式P1**

**用户查询B2，要把数据库的查询index换为多项式的index（P1）**

**服务端返回P1的加密**

**客户端同态解密，得到明文多项式，根据pack的偏移找到B2对应的系数，拼接为明文**

1. 查询向量压缩为一个密文（显著减少通信量）

**查询向量：{0…1…0}只有一位为1**

**HE明文：Σ ai \* x^I，要查的内容ai为1（i范围是0到N-1）**

**加密明文得到密文**

**服务端使用扩展算法得到N个具体密文向量，执行同态乘就可以得到加密结果**

1. 支持多维查询

**以二维查询为案例，查询向量N维数据转为√n \* √n的矩阵（明文）**

**。。。(看不懂了，跳过)**

**客户端最后收到一些密文向量，解密后再组合成密文，然后再次解密**

1. 支持多个查询（cuckoo hash）

**待查询的多个数据插入simple hash**

**服务端把index插入simple hash**

**用户对每一行（bin）查询，可以得到多个查询**

## sf中的实现位置

（可能已经更改，这里是更改前的位置）

实现：

libspu/pir/seal\_pir.\*

libspu/pir/seal\_mpir.\*

外部依赖：

bazel/seal.BUILD

bazel/repositories.bzl

单元测试：

libspu/pir/seal\_pir\_test.cc

libspu/pir/seal\_mpir\_test.cc

# Keyword PIR – Labeled PSI介绍

## 基本原理

使用两个插值多项式

一个是匹配多项式P，一个是label插值多项式Q

当Pk=0，对应的Qk的值就是待查询的value

这种方式同样使用BFV方案，但是明文编排的方式采用batch encode，支持SIMD(每个位置上的数字可以做对应的运算)

论文思想：

1. cuckoo希降低差值多项式的次数(CLR17)
2. 引入oprf避免信息泄露。(CLHR18)
3. 一系列优化和开源实现(CMCD+21)

论文的优化点：

一、减少乘法次数和计算量的方法

1. 客户端可以通过窗口机制。发送一个数的n次方（1，2，4…）
2. 服务器通过分区的方式，把一个bin划分为一些子集，每个子集对应一个差值多项式，降低多项式次数

二、使用external postage stamp bases降低乘法深度，减少通信量。

三、使用Paterson-Stockmeyer降低多项式次数

## sf label PSI的工作

### 主要实现内容

1. 封装微软的开源代码
2. OPRF采用sf自己的实现，支持以下ecc曲线：FourQ，Secp256k1，SM2
3. 使用yacl提供的link，包含对label的自动填充
4. 增加了服务端的预处理结果的保存功能，支持离线建立和在线查询（多次）两个阶段

### 执行流程

服务端处理阶段(setup)

1. 选择参数，包括cuckoo hash、同态
2. 对服务端的数据进行prf计算，一共会得到256位数字。前128比特截取用于支持匹配，后128比特呢用于加密label。
3. 把prf 得到的前128位插入simple hash。
4. 对Simple hatch的每一行分别进行分桶处理，计算两个差值多项式
5. 把插值多项式的系数pack到同态算法明文

客户端和服务器通信阶段(query)

1. 客户端向服务器请求相关参数
2. 执行oprf协议
3. 计算查询值的同态密文幂的集合
4. 使用同态加密的私钥解密服务端返回的密文
5. 匹配条件满足时使用oprf的后128位解密得到label

### 相关文件

（在成为独立的代码库之前是这样的，现在不知道在哪里）

实现：

libspu/psi/core/labeled\_psi/\*

libspu/pir/pir.\*

外部依赖：

bazel/repositories.bzl

bazel/Microsoft\_apis.BUILD

…

bazel/patches/apis.patch

# PIR后续计划

一、pir协议开发

sota pir跟踪，提升pir性能（如2022年的spiral pir和2023年的simple pir）

二、pir调用框架

构建psi和pir的独立代码库

三、产品化

了解产品需求

设计落地方案