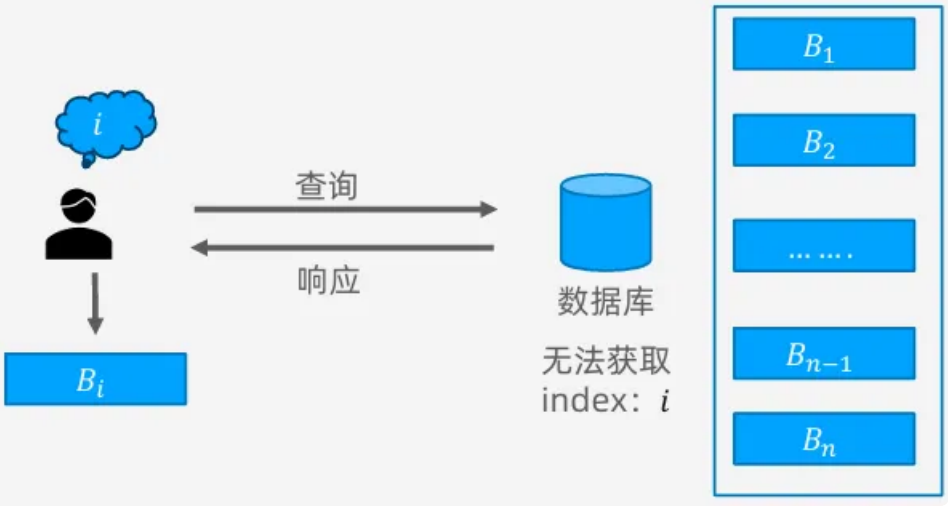
第6讲 隐语PIR功能及使用介绍

隐语实现PIR总体介绍

见第2讲笔记

用户查询服务端数据库中的数据，
但服务端不知道用户查询的是哪些数据



按服务器数量分类：

- 单服务器方案（Single Server）
- 多服务器方案（Multi-Server）

按查询类型分类：

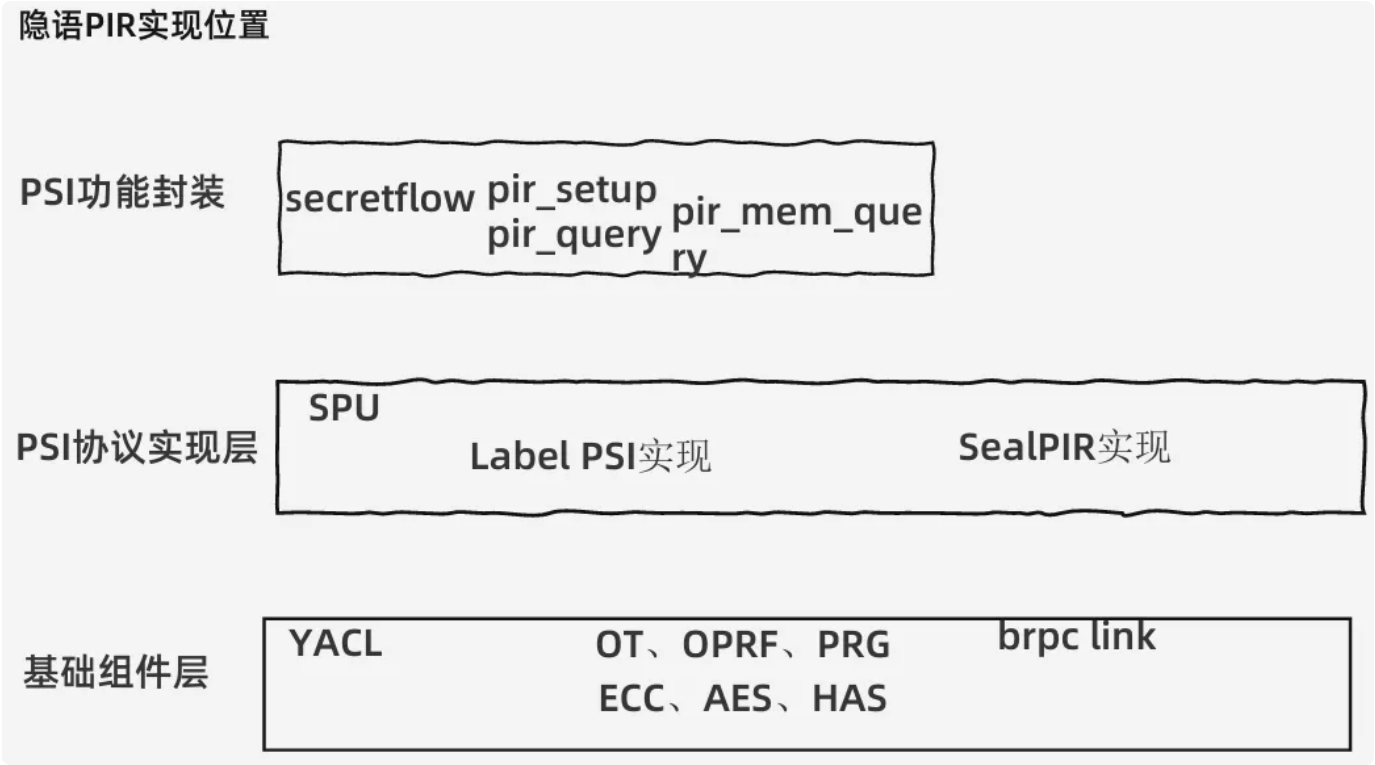
- Index PIR
- Keyword PIR

隐语目前支持的PIR方式：

Single Server Index PIR： SealPIR

Single Server Keyword PIR：Labeled PSI

实现位置



调用接口 pir_setup

隐语PIR 调用接口：pir_setup

pir_setup 数据预处理，参数说明：

- input_path

服务端数据文件路径，建议绝对路径

- key_columns

Key对应的列名

- label_columns

Label对应的列名，多列，用逗号分隔

- oprf_key_path

服务端ecc密钥文件，32B，二进制文件

- mum_per_query

每次查询的id数量

- label_max_len

Label数据拼接后填充到固定的长度大小

```
reports = spu.pir_setup(
    server='bob',
    input_path='/path/B_PIR_DATA.csv',
    key_columns='id',
    label_columns=['register_date','age'],
    oprf_key_path='/path/oprf_key.bin',
    setup_path='/path/setup_path',
    num_per_query=1,
    label_max_len=18,
)
```

提示：pir_setup 单方任务：

可以使用的secretflow单机模拟配置，或者直接调用spu的python接口

调用接口 pir_query

隐语PIR调用接口：pir_query

pir_query双方执行查询任务, 参数说明：

- 客户端配置
  - input_path
    查询id对应的csv文件路径
  - Key_columns
    Key对应的列名
  - output_path
    PIR查询结果输出的文件路径
- 服务端配置
  - oprf_key_path
    服务端ecc密钥文件，32B，二进制文件
  - setup_path
    预处理阶段结果输出路径

```
# client
alice_config = {
    'input_path': '/path/A_PIR_ID.csv',
    'key_columns': id',
    'output_path': '/path/sf_pir_out.csv',
}
# server
bob_config = {
    'oprf_key_path': '/path/oprf_key.bin',
    'setup_path': '/path/setup_path',
}
query_config = {
     alice: alice_config,
     bob: bob_config,
}

reports = spu.pir_query(
    server='bob',
    config=query_config,
)
```

Index PIR-SealPIR介绍

BFV方案

BFV方案介绍

- 参数
  - 多项式次数: N
  - 明文模：t
  - 密文模：q
  - Expansion Rate

$$2 * \log(q)/\log(t)$$

- 明文

$$R_t = \mathbb{Z}_t[x]/(x^N + 1)$$

$$a_0 + a_1 x + \cdots + a_{N-2} x^{N-2} + a_{N-1} x^{N-1},$$

- 密文$(c_0, c_1) \in R_q \times R_q$

$$R_q = \mathbb{Z}_q[x]/(x^N + 1)$$

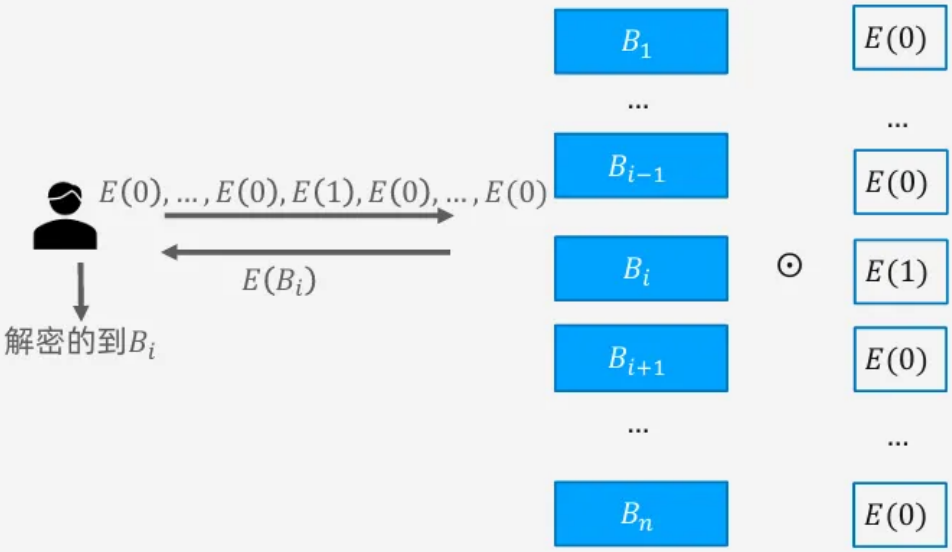
- 密文加法：明文$p_1(x), p_2(x)$的密文 c_1, c_2, $c_1 + c_2$ 是$p_1(x) + p_2(x)$的密文
- 明文乘密文：明文$p_1(x)$的密文 c，$p_2(x) \cdot c$是$p_1(x) \cdot p_2(x)$的密文
- 替换：明文$p(x)$的密文 c，奇数k

$Sub(c, k)$是$p(x^k)$ 的密文，例如$p(x) = 7 + x^2 + 2x^3$

$Sub(c, 3)$，得到$p(x^3) = 7 + (x^3)^2 + 2(x^3)^3 = 7 + x^6 + 2x^9$的密文

operation	CPU cost (ms)	noise growth
addition	0.002	additive
plaintext multiplication	0.141	multiplicative*
multiplication	1.514	multiplicative
substitution	0.279	additive

基于同态密码实现PIR基本原理



方案的问题：查询请求n个密文向量，太大

主要贡献

- 多个数据pack到一个HE Plaintext

  查询的db_index转换为plaintext_index

- 查询向量压缩到一个密文

  显著减少通信量，server端可通过计算expand得到查询密文向量

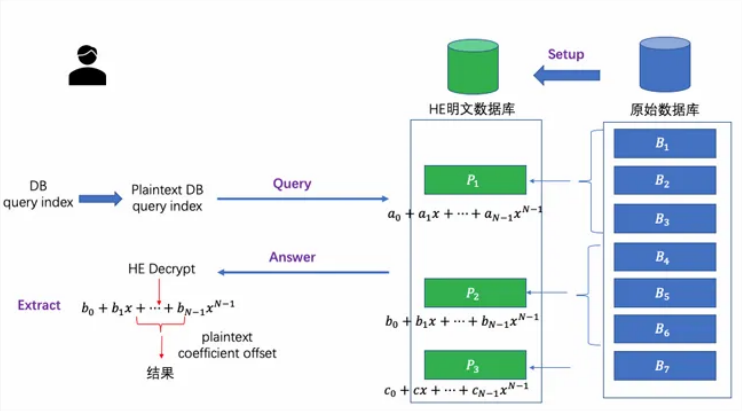
- 支持多维查询

  2维查询将数据转换为$\sqrt{n} * \sqrt{n}$的矩阵，减少expand计算量

- 支持多个查询

  使用cuckoo hash支持同时进行多个查询

多个数据pack到一个HE plaintext:



多项式次数：8192

明文模：17bit

DB数据长度：288B

Ceil(288*8/16) = 144

Floor(8192/144) = 56

每个多项式可以pack 56个原始数据

备注：实现Symmetric PIR时，不使用packing

查询向量压缩:

- 客户端生成查询密文

查询向量：{0,0, … , 1, … , 0}转换为HE 明文plaintext

$$a_0 + a_1 \cdot x + \cdots + a_{N-1} \cdot x^{N-1} = x^{query\_indx}$$

$$a_i = 0, i \neq query\_index$$

$$a_i = 1, i = query\_index$$

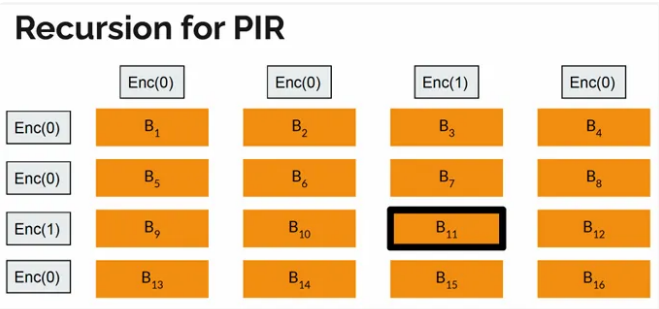
$$加密 Enc\left(a_0 + a_1 \cdot x + \cdots + a_{N-1} \cdot x^{N-1}\right) = Enc(x^{query\_indx})$$

- 服务端通过执行Expand算法，得到密文向量:

$$Enc(0), Enc(0), \ldots, Enc(1), \ldots, Enc(0)$$

备注:
每个明文可以pack多个(56)原始数据
通过一个查询密文，可以对8192*56=458,752个原始数据进行查询

- 多维查询，减少查询向量



1. $\sqrt{n} * \sqrt{n}$的矩阵M，其中$\sqrt{n} \leq 8192$
2. V_c是密文查询列向量， $A_c = M \cdot V_c$， A_c是$\sqrt{n} * 1$的密文列向量
3. 将A_c中每个密文拆分为F份，得到$\sqrt{n} * F$的明文矩阵AS_c
4. V_r是密文查询行向量， $AS_c^T \cdot V_r$，得到$F * 1$ 的密文向量
5. 客户端收得到F个密文向量，解密后，将其组合成密文 $Enc(M[r, c])$，再次解密得到$M[r, c]$

备注：

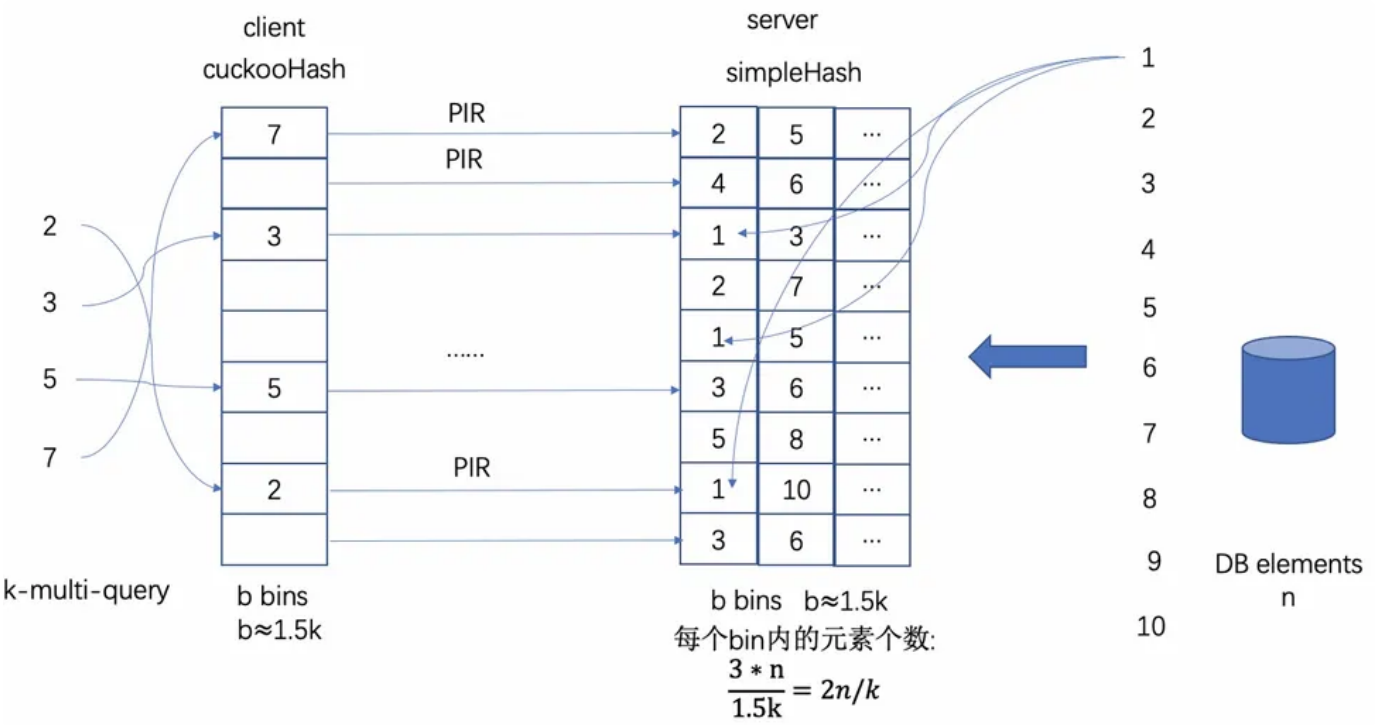
$F = 2 * \log(q)/\log(t)$，是扩张因子

$F = 2 * 218/16 \approx 28$

密文扩展： F^2

- 8192*8192*56=3,758,096,384≈37亿的数据

- 支持多个查询



SealPIR 中给出了通过cuckooHash进行多个查询的算法，可以提高查询效率。cuckooHash选取3个hash，bin的个数为1.5k，具体算法，如下：

server setup

- 对DB中n个元素，分别计算cuckooHash的3个Hash，得到3个bin index，插入到3个bin index中

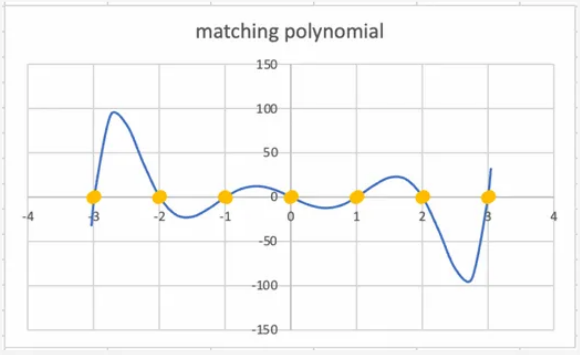
client query

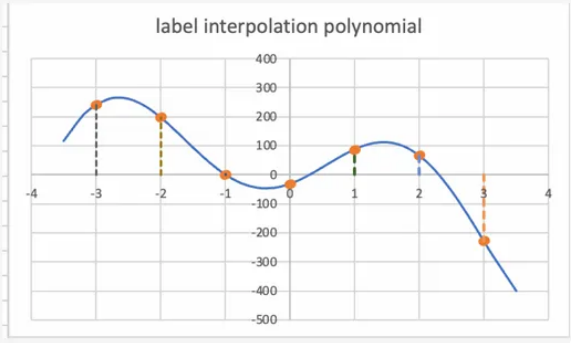
- 将k个查询，通过cuckooHash插入到b=1.5k个bin中，对空的bin进行随机填充，
- 执行每个bin执行一个PIR，共计b个PIR。这里的难点是如何将query_index转换为bin_index

Keyword PIR-Labeled PSI介绍

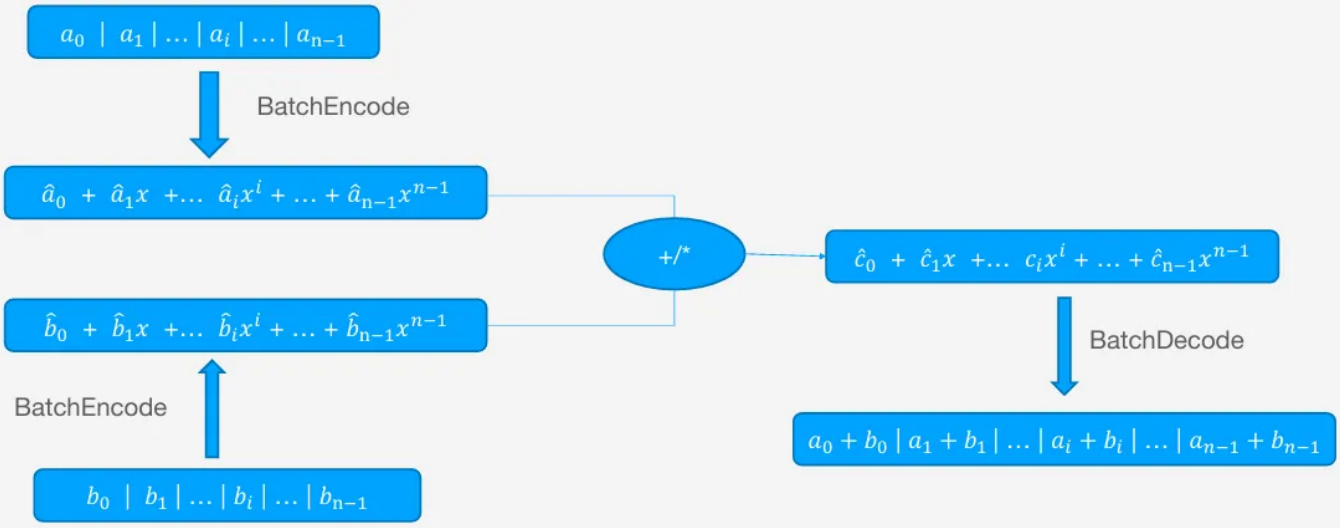
基本原理

核心思想：点值表示得到插值多项式系数表示





BFV SHE: packing and SIMD



SEAL 编译选项：
SEAL_THROW_ON_TRANSPARENT_CIPHERTEXT设为OFF

减少乘法次数和计算量

- 窗口(windowing)-发送$[\![r]\!]$的次幂，减少密文乘法

Receiver: $[\![r]\!], [\![r]\!]^2, [\![r]\!]^4, \ldots [\![r]\!]^{2^{\lfloor log_2 N_S \rfloor}}$

- 分区(partitioning)

Sender 每个bin划分成若干subset，bundle

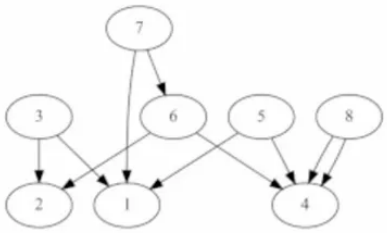
每个partition对应一个多项式

Sender: $P_{k,0}(x) = \sum_{i=0}^{3} P_{k,i} x^i \qquad P_{k,1}(x) = \sum_{i=4}^{7} P_{k,i} x^i$

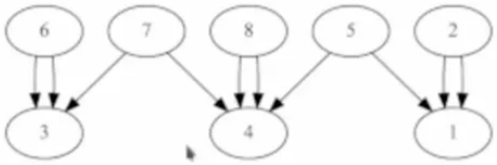
Receiver: $P_{k,0}(r)$ $P_{k,1}(r)$ 有一个为零时，即表示$r \in S$

使用extremal postage stamp bases减少通信量

• 给个k个正整数$A_k = \{a_1 = 1 < a_2 < \cdots < a_k\}$，$[1, n]$中的所有整数可以有最多h个$a_j$相加得到。



Windowing: {1, 2, 4}，乘法深度2



postage stamp bases，{1, 3, 4}，乘法深度1

Paterson–Stockmeyer算法，减少密文乘法

- $P([\![r]\!]) = \sum_{i=0}^{H-1} [\![r]\!]^{iL} \sum_{j=0}^{L-1} a_{iL+j} [\![r]\!]^{j}$

示例

$\mathrm{M}(x) = 1 + 2x + 3x^2 + 4x^3 + 5x^4 + 6x^5 + 7x^6 + 8x^7 + 9x^8 + 10x^9 + 11x^{10} + 12x^{11} + 13x^{12} + 14x^{13} + 15x^{14}$

$L = 4, \mathrm{H} = 4$

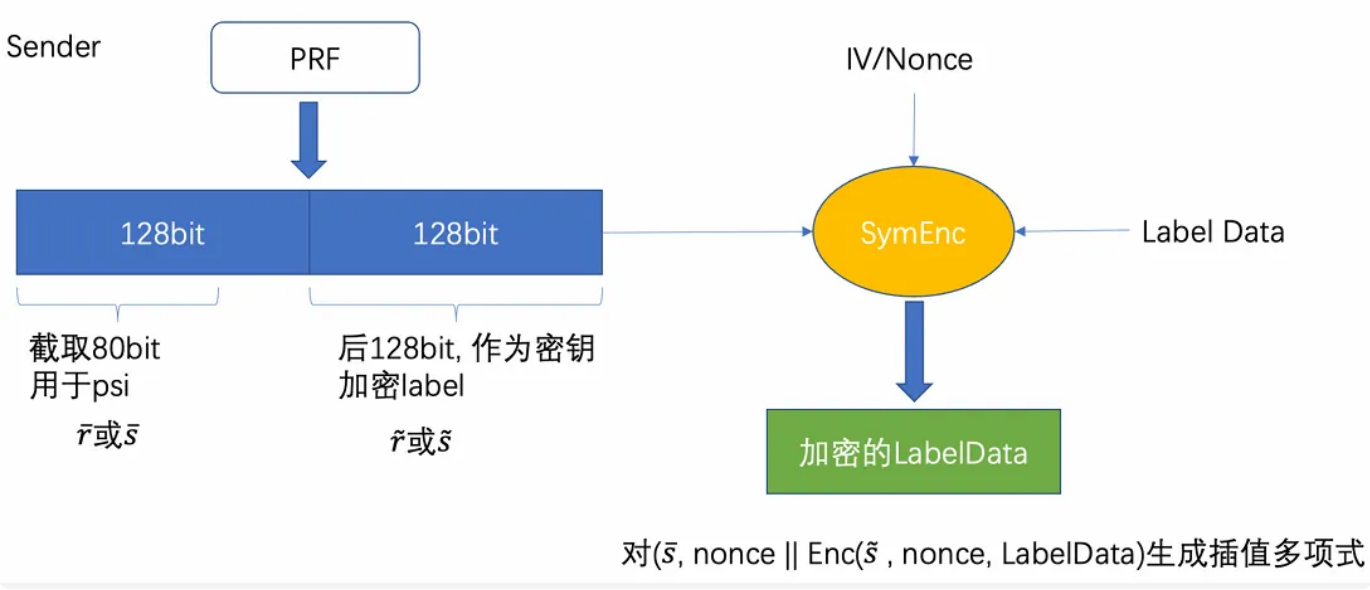
- $\mathrm{M}(x) = 1 + 2x + 3x^2 + 4x^3$
  $+x^4(5 + 6x + 7x^2 + 8x^3)$
  $+x^8(9 + 10x + 11x^2 + 12x^3)$
  $+x^{12}(13 + 14x + 15x^2 + 0x^3)$

query_powers={1,2,3,4,8,12}

隐语label PSI的主要工作

- 以微软的开源代码功能为核心
- OPRF采用隐语的实现：
  - 支持的ecc曲线包括：FourQ, Secp256k1, SM2
- Label的自动填充
- 增加了服务的预处理结果保存功能，
  - 可以支持离线和查询(多次)两个阶段

服务端预处理(setup)阶段–流程



1. 选择参数
2. 对id数据进行prf计算，前128bit根据截取用于匹配，后128bit作为对称算法密钥加密label
3. 根据prf前128bit将数据插入Simple Hash
4. 对Simple Hash每一行分别划分bin bundle，并计算matching polynomial和label polynomial
5. 将插值多项式系统packing到同态算法明文

客户端和服务端(query)阶段–流程

	Client(Receiver)		Server(Sender)
		offline	
	client	online	server Setup
step1	Request Params	⟶	
		⟵	Response Params
step2	Request OPRF	⟶	
		⟵	Response OPRF
step3	Request Query	⟶	
		⟵	Response Query

1. 请求参数
2. 执行oprf协议
3. 计算查询值的同态密文幂集合
4. 使用同态私钥解密服务端返回的同态密文
5. 满足匹配条件时，使用oprf的后128bit解密得到label

主要工作

参数大类	参数名称	参数说明
ItemParams		
	felts_per_item	item_bit_size / plain_modulus_bits item_bit_size = stats_params + log(ns)+log(nr)
TableParams		
	hash_func_count	cuckoo的相关参数
	table_size	
	max_items_per_bin	
QueryParams		
	ps_low_degree	为了降低发送密文$[\![x^i]\!]$个数，以及服务端密文乘法的次数的一些优化参数
	query_powers	
SEALParams		
	poly_modulus_degree	2048 / 4096 / 8192
	plain_modulus(_bits)	65535 / 65535 / 22(bits)
	coeff_modulus_bits	{48} / {48, 30, 30} / {56, 56, 56, 50}

隐语PIR后续计划

PIR协议开发

- SOTA PIR跟踪
  - Spiral PIR(2022)
  - Simple PIR(2023)

PIR调用框架

- PSI/PIR独立代码库

PIR产品化

- 了解产品需求
- 设计落地方案