# 联　　盟　　规　　范

PPCA 7 —2023

# 隐私计算 跨平台互联互通 开放协议 第 2 部分：SS-LR

Privacy-preserving computation cross-platform interconnection—

Open protocol Part2: Secret Sharing based Logistic Regression protocol

2023-07-07 发布　　2023-07-07 实施

隐私计算联盟 发 布

# 目　次

# 前　言

本文件是《隐私计算 跨平台互联互通》系列文件之一，该系列文件名称如下：

——开放协议 第1部分：ECDH-PSI；

——开放协议 第2部分：SS-LR。

请注意本文件的某些内容可能涉及专利。本文件的发布机构不承担识别这些专利的责任。

本文件由隐私计算联盟提出并归口。

本文件起草单位：中国信息通信研究院、蚂蚁科技集团股份有限公司、中移动信息技术有限公司、联通数字科技有限公司、天翼电子商务有限公司、深圳市洞见智慧科技有限公司。

本文件主要起草人：邵健、白玉真、陈卓、孙林、贺伟、何浩、茹志强、昌文婷、陆宇飞、袁博、章庆、靳新、余超凡、冯玲云、马永刚、袁鹏程、苏亮、赵原、李漓春、张涛、徐文静、王思源、张启超、张晓蒙、张鸣皓。

# 引　言

当前多方安全计算、联邦学习等隐私计算技术快速发展，越来越多的产品从试点部署阶段转入落地应用，市场竞争火热。但是，不同技术厂商提供的产品和解决方案在设计原理和功能实现之间存在较大差异，使得部署于不同平台的隐私计算参与方之间无法跨平台完成同一计算任务，为实现与部署于不同平台的多个合作方之间的数据融合，用户往往不得不部署多套产品以逐一适配。作为促进跨机构间数据共享融合的关键技术，隐私计算有望成为支撑数据流通产业的基础设施，但高额的应用成本不利于隐私计算技术的推广应用。因此，解决不同产品之间的技术壁垒，实现计算任务在跨平台间的互联互通已成为产业内的迫切需求。

# 版权声明



邮箱：ppca@caictyds.cn

# 隐私计算 跨平台互联互通
# 开放协议 第2部分：SS-LR

## 1 范围

本文件规定了隐私计算平台间的基于秘密分享的逻辑回归互联互通的算法协议和参考实现。

## 2 规范性引用文件

下列文件中的内容通过文中的规范性引用而构成本文件必不可少的条款。其中，注日期的引用文件，仅该日期对应的版本适用于本文件；不注日期的引用文件，其最新版本（包括所有的修改单）适用于本文件。

## 3 术语和定义

GB/T 25069-2022界定的以及下列术语和定义适用于本文件。

3.1

**隐私计算 privacy-preserving computation**

在保证数据提供方不泄露原始数据的前提下，对数据进行分析计算的一系列信息技术，保障数据在流通与融合过程中的“可用不可见”。

3.2

**开放协议 open protocol**

通过规范算法执行流程中交互信息，各平台独立开发算法来实现平台间算法的互通。

3.3

**节点 node**

各隐私计算技术平台的部署实例，是互联互通网络的基本组成单元，对外提供交互接口。

3.4

**算法 algorithm**

为解决问题严格定义的有限的有序规则集。

[来源：GB/T 25069—2022, 3.581]

3.5

**组件 component**

独立执行隐私计算任务的模块单元，其经过封装、符合开放接口规范、可以完成某个特定计算或算法，可独立部署。

3.6

**任务** task
组件运行实例的载体。

## 4 缩略语

下列缩略语适用于本文件。

| | | |
|---|---|---|
| LR | 逻辑回归 | Logistic Regression |
| SS | 秘密分享 | Secret Sharing |

## 5 算法协议

### 5.1 LR 算法训练过程

LR 算法公式为：

$$y = sigmoid(x_0 w_0 + x_1 w_1 + \cdots + x_k w_k + intercept)$$

其中 x 表示特征项，w 是 LR 模型的权重，intercept 是截距项， w 和 intercept 都是可训练的参数，sigmoid 是一个 S 型函数，公式如下：

$$sigmoid(x) = \frac{1}{1 + e^{-x}}$$

在隐私计算场景下，Sigmoid 函数直接计算有困难，本文件使用近似函数计算的方案，详见 5.2.3.1 节。

此外，为了统一 w 和 intercept 的形式，简化计算流程，我们将 intercept 也看作是 w 向量的一个元素，即假设特征数为 k 个，则 w 向量的长度为 k+1，之后在计算时，我们在 x 的最后拼接一个 1，即可直接计算向量内积 w*x，简化计算步骤。

在实际训练时，LR 模型训练采用 mini-batch 梯度下降方法，重复执行如下五个步骤，直到达到目标迭代次数：

第一步：计算预测值 $pred = sigmoid(batch\_x * w)$

第二步：计算误差 $err = pred - y$

第三步：计算梯度 $grad = transpose(batch\_x) * err$

第四步：梯度加正则项 $grad = grad + w' * l2\_norm$

第五步：更新模型参数 $w = w - (grad * learning\_rate / batch\_size)$

其中：

- $batch_x$是当前 batch 的样本数据集与常数项组成的矩阵,其中常数项为 1，位于$batch_x$的最后一列；$w$是 LR 模型训练的结果，初始值为$\mathbf{0}$；$sigmoid()$是函数常被用作神经网络的激活函数，将变量映射到 0,1 之间；$*$是乘法；
- $y$是当前 batch 的标签矩阵；
- $transpose()$是矩阵转置操作；

- $w'$是将$w$的截距项即最后一个元素置为 0；$l2_norm$是$l2$正则项；
- $learning_rate$是学习率参数；$batch_size$是每个 batch 的大小。

## 5.2 基于 Semi2K 的纵向 SS-LR 算法

本节描述基于Semi2K协议的两方秘密分享纵向LR算法的计算过程。

### 5.2.1 数据集初始化

各参与方首先对输入的样本数据$batch_x$和标签矩阵$y$进行编码和秘密分片。

#### 5.2.1.1 输入数据编码

输入数据集的元素类型为浮点数类型，将其按统一的放大系数转为定点数，然后去掉小数部分转为整数，在整数环上计算。编码方式如表 1 所示，整数环可取长度 64、128 等，元素编码后的数据类型分别为 int64_t、int128_t。放大系数和整数环的比特数在握手协议中约定。

表 1 输入数据集元素编码

| 实际数据类型 | 整数环的比特数 | 编码数据类型（ISO C99） |
|---|---|---|
| 浮点数 | 64 | int64_t |
| 浮点数 | 128 | int128_t |

#### 5.2.1.2 输入数据秘密分享

双方根据握手协议确定的样本数量、特征数量和标签信息，以及各自的输入样本数据集。首先通过秘密分享的方式计算出初始密态样本数据集。对于每个样本元素$x$，秘密分享获得两个密态分片 $x$和 0（$x = x + 0$），将$x$留在本地，将 0 发给对方（由于 0 不是随机数，所以省略这个发送过程），示例如附录 A 所示。

#### 5.2.1.3 交换伪随机数种子

双方各有一个属于自己的伪随机数种子，用于将公共明文转为密文，详见5.2.3.2.1节。初始阶段双方向对方发送这个伪随机数种子。

#### 5.2.1.4 向 Beaver 服务发送伪随机数种子

除了5.2.1.3节的伪随机数种子，双方还各自持有另一个伪随机数种子，用于从第三方可信 Beaver 服务获取乘法三元组，详见5.3.2节和附录B。初始阶段双方分别向Beaver服务发送自己的这个伪随机数种子。

### 5.2.2 伪随机数生成方式

密码学安全的伪随机数生成器应当按照美国 NIST SP800-90A 标准或者中华人民共和国密码行业标准《GM/T 0105-2021 软件随机数发生器设计指南》中规定的方式实现。伪随机数的安全强度要达到 128 比特。

#### 5.2.3 安全算子

本节定义安全算子的计算方式。LR 算法运行的复杂算子包含 Sigmoid 算子。

##### 5.2.3.1 Sigmoid 算子

在 Semi2K 协议下无法直接计算 Sigmoid 函数，本文件使用近似计算方案。Sigmoid 有多种拟合算法，其中一种较为简单，并且不需要交互的近似计算方法如下：

Minimax 逼近方法，其公式为：$f(X) = 0.5 + 0.125 * X$。

##### 5.2.3.2 Semi2k 算子

###### 5.2.3.2.1 公共明文转为密文

将$batch_x$常数项1转为密文，将$w'$的截距项0转为密文，这两处涉及公共明文转为密文。

假设公共密文等于$p$，参与方A使用自己的伪随机数种子生成随机数$r_0$，B使用自己的伪随机数种子生成随机数$r_1$，生成方式参照5.2.2节。由于双方都知道对方的伪随机数种子，且双方维护的计数器始终保持一致，所以可以计算出对方生成的伪随机数。最终，A的密态分片等于$p + r_0 - r_1$，B的密态分片等于$r_1 - r_0$。

###### 5.2.3.2.2 明文常量 × 密文向量

Sigmoid 的 Minimax 逼近公式中涉及明文常量与密文向量相乘，即$0.125 * X$，$w' * l2\_norm$。

此算子不涉及参与方之间的通信，双方各自在本地将明文常量与密文分片直接相乘即可。

###### 5.2.3.2.3 明文常量 + 密文向量

Sigmoid 的 Minimax 逼近公式中涉及明文常量与密文向量相加，即$0.5 + 0.125 * X$。

对于两个参与方（参与方 A 为 rank 0，参与方 B 为 rank 1）的情况，将明文常量直接加在 A 方的密文分片上，B 方的密文分片不变。

###### 5.2.3.2.4 密文向量 + 密文向量

梯度更新公式 $grad = grad + w' * l2\_norm$ 涉及密文向量与密文向量相加。

此算子不涉及参与方之间的通信，双方在本地将两个密文的分片直接相加即可。

###### 5.2.3.2.5 密文向量 - 密文向量

误差计算公式 $err = pred - y$ 涉及密文向量与密文向量相减。此算子不涉及参与方之间的通信，双方在本地将两个密文的分片直接相减即可。

###### 5.2.3.2.6 密文矩阵 * 密文矩阵

公式 $batch\_x * w$ 和公式 $grad = transpose(batch\_x) * err$ 涉及密文矩阵与密文矩阵相乘。算子的实现定义如下：

$X_i$、$Y_i$ $(i = 0\ or\ 1)$ 分别是 $X$、$Y$ 的密文分片，以下步骤是计算 $Z_i = [X * Y]_i$ 的实现。

(1) 可信第三方生成 Beaver 矩阵乘法三元组$(A_i\ \ B_i\ \ C_i)$，生成方式参见 5.3.2 节。

(2) 双方同时解密得到 $X - A$ 和 $Y - B$：

双方分别本地计算 $X_i - A_i$ 和 $Y_i - B_i$ ，然后调用传输层协议接口先后获得对方的分片 $X_{1-i} - A_{1-i}$ 和 $Y_{1-i} - B_{1-i}$。双方分别在本地执行加法后得到 $X - A$ 和 $Y - B$。

(3) 各方分别在本地执行以下计算：

$$Z_i = C_i + (X - A) \cdot B_i + A_i \cdot (Y - B) + (1 - i) \times (X - A) \cdot (Y - B)$$

注：对于初始计算 $batch_x * w$时，$w$等于 0，各参与方在本地计算计算$batch_x * w$即可。

(4) 对$Z_i$进行截断，截断的实现方式参照 5.2.3.2.7 节。

#### 5.2.3.2.7 密文截断

乘法计算后要对结果进行截断（Truncation）。有多种对秘密分享分片截断的方案。

其中一种不需要交互但有一定概率出错的截断算法方案如下：

一方直接对其分片向右移动指定截断位数，另一方用$2^n$减分片，将减法结果向右移动指定截断位数，再用$2^n$减右移后的分片。

此处的移动指逻辑右移，即对于 int64/int128 变量，如果数值为自然数，右移时高位补 0，如果数值为负数，右移时高位补 1。

### 5.3 Beaver 矩阵乘法三元组

#### 5.3.1 概述

在矩阵乘法计算过程中，参与方 A 和 B 获得矩阵乘法三元组分片（$A_0$，$B_0$，$C_0$）和（$A_1$、$B_1$、$C_1$），满足$(A_0 + A_1) \cdot (B_0 + B_1) = (C_0 + C_1)$。三元组分片是各方通过私有的随机数种子生成的，所以对另一方不可见。

#### 5.3.2 Beaver 三元组

乘法三元组有多种实现算法方案，包括两方实现的算法方案和第三方辅助的算法方案。以下是一种第三方辅助的算法方案的实现设计。

本小节以可信第三方的 Beaver 服务为例。Beaver 服务器知道参与方 A 和参与方 B 的随机数种子，通过相同的伪随机数生成算法和参数生成与 A 和 B 相同的伪随机数，即三元组分片分片（$A_0$，$B_0$，$C_0$）和（$A_1$、$B_1$、$C_1$）。Beaver 服务计算 $adjust = (A_0 + A_1) \cdot (B_0 + B_1) - (C_0 + C_1)$，得到调整值 $adjust$。参与方 A（或参与方 B）从 Beaver 服务获取$adjust$值，并将其加到$C_0$（或$C_1$）分片上得到新的$C_0$（或$C_1$），最终得到的就是符合条件的矩阵乘法三元组。基于可信第三方的 Beaver 服务是一个 RPC 服务，接口协议采用 ProtoBuf 协议，如附录 B 所示。

## 6 算法流程

算法流程包含两个阶段，如图1。第一阶段为算法协商握手阶段，第二阶段为算法主体运行阶段：

a） 算法协商握手阶段，确定算法版本、算法参数、密码协议及相关参数、安全算子及相关参数、输入数据集参数等算法运行所需的信息；
b） 算法主体运行阶段，进行LR模型训练，计算出模型参数$w$。

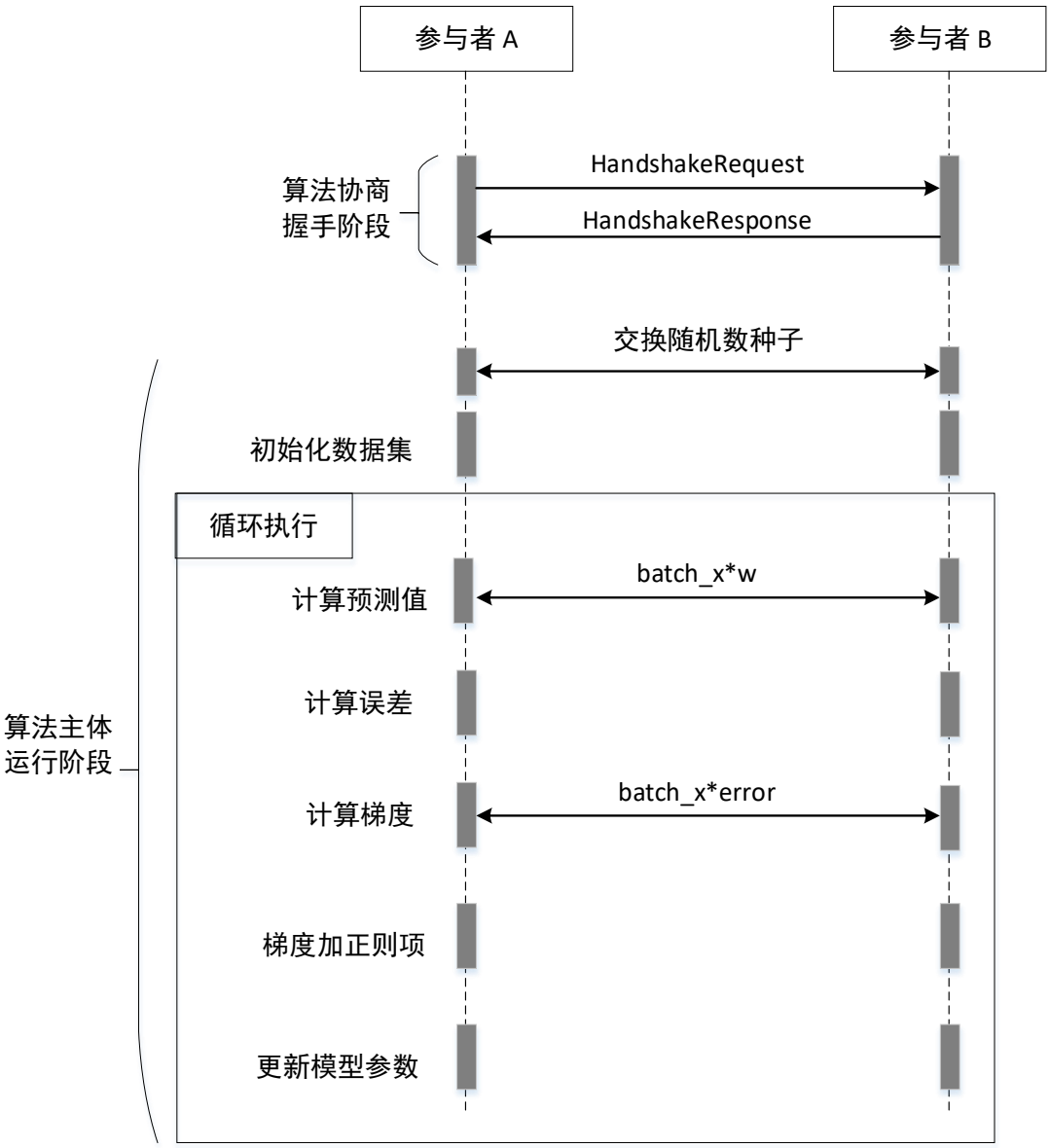


图 1 SS-LR 协议流程

## 7 算法协商

### 7.1 HandshakeRequest 消息

HandshakeRequest包括算法协商握手请求的基本信息，其数据结构如表2所示。

表 2 HandshakeRequest 数据结构

| 属性名称 | 数据类型 | 数据说明 | 示例 | 数据备注 |
|---|---|---|---|---|
| version | int32 | 握手请求版本号 | 2 | 必选 |

| 属性名称 | 数据类型 | 数据说明 | 示例 | 数据备注 |
|---|---|---|---|---|
| requester_rank | int32 | 发送方 rank 值 | 1 | 必选 |
| supported_algos | int32 list | 支持的开放协议的 enum 值，如 SS-LR | 见表 3 | 必选 |
| algo_params | google.protobuf.Any list | 相应的算法详细握手参数，与 supported_algo 对应。LR 算法的类型是 LrHyperparamsProposal。 | 见表 4 | 必选 |
| ops | int32 | 复杂算子对应的 enum 值，如 sigmoid 算子 | 见表 7 | 必选 |
| op_params | google.protobuf.Any list | 相应的复杂算子的详细握手参数，与 ops 对应。实际类型随复杂算子类型而变，sigmoid 算子的类型是 SigmoidParamsProposal。 | 见表 8 | 必选 |
| protocol_families | int32 list | 支持的协议族的 enum 值，如 SS 协议族 | 见表 10 | 必选 |
| protocol_family_params | google.protobuf.Any list | 相应的协议族详细握手参数，与 protocol_families 对应。实际类型随协议族类型而变，SS 协议族的类型是 SSProtocolProposal | 见表 11 | 必选 |
| io_param | google.protobuf.Any | LR 算法的输入和结果输出格式参数，与 algo 对应。<br>实际类型随算法类型而变，SS-LR 协议的类型是 LrDataIoProposal | 见表 20 | 必选 |

其中：

- supported_algos列表元素的取值和说明如表3所示。

表 3 supported_algos 的值域表

| 取值 | 数据说明 |
|---|---|
| 1 | ALGO_TYPE_ECDH_PSI，ECDH-PSI 算法 |
| 2 | ALGO_TYPE_SS_LR，SS-LR 算法 |

- LrHyperparamsProposal包括SS-LR算法参数协商的基本信息，如表4所示。

表 4 LrHyperparamsProposal 数据结构

| 属性名称 | 数据类型 | 数据说明 | 示例 | 数据备注 |
|---|---|---|---|---|
| supported_versions | int32 list | 支持的版本列表 | [1] | 必选 |
| optimizers | int32 list | 支持的迭代优化器类型列表 | 见表 5 | 必选 |
| last_batch_policies | int32 list | mini-batch 梯度下降法处理最后一个 batch 的策略的列表，如 discard | 见表 6 | 必选 |
| use_l0_norm | bool | 是否采用 l0 正则 | false | 必选 |

| 属性名称 | 数据类型 | 数据说明 | 示例 | 数据备注 |
|---|---|---|---|---|
| use_l1_norm | bool | 是否采用 l1 正则 | false | 必选 |
| use_l2_norm | bool | 是否采用 l2 正则 | true | 必选 |

- optimizers元素的取值和说明如表5所示。

表 5 optimizers 元素的值域表

| 取值 | 数据说明 |
|---|---|
| 1 | OPTIMIZER_SGD，SGD 方法 |
| 2 | OPTIMIZER_ MOMENTUM，Momentum 方法 |
| 3 | OPTIMIZER_ ADAGRAD，Adagrad 方法 |
| 4 | OPTIMIZER_ ADADELTA，Adadelta 方法 |
| 5 | OPTIMIZER_ RMSPROP，RMSprop 方法 |
| 6 | OPTIMIZER_ ADAM，Adam 方法 |
| 7 | OPTIMIZER_ ADAMAX，AdaMax 方法 |
| 8 | OPTIMIZER_ NADAM，Nadam 方法 |

- last_batch_policies元素的取值和说明如表6所示。

表 6 last_batch_policies 元素的值域表

| 取值 | 数据说明 |
|---|---|
| 1 | LAST_BATCH_POLICY_DISCARD，丢弃最后一个不完整的 batch |

- ops列表元素的取值和说明如表7所示。

表 7 ops 元素的值域表

| 取值 | 数据说明 |
|---|---|
| 1 | OP_TYPE_SIGMOID，SIGMOID 算子 |

- SigmoidParamsProposal包括sigmoid算子参数协商的基本信息，如表8所示。

表 8 SigmoidParamsProposal 数据结构

| 属性名称 | 数据类型 | 数据说明 | 示例 | 数据备注 |
|---|---|---|---|---|
| supported_versions | int32 list | 支持的版本列表 | [1] | 必选 |
| sigmoid_modes | int32 list | sigmoid 逼近公式类型的 enum 值，如如 minimax | 见表 9 | 必选 |

- sigmoid_modes元素的取值和说明如表9所示。

表 9 sigmoid_modes 元素的值域表

| 取值 | 数据说明 |
| --- | --- |
| 1 | SIGMOID_MODE_MINIMAX_1，minimax 一阶近似公式 |

- protocol_families元素的取值和说明如表10所示。

表 10 protocol_families 元素的值域表

| 取值 | 数据说明 |
| --- | --- |
| 1 | PROTOCOL_FAMILY_ECC，PSI 协议族 |
| 2 | PROTOCOL_FAMILY_SS，SS 协议族 |
| 3 | PROTOCOL_FAMILY_PHE，PHE 协议族 |

- SSProtocolProposal包括SS协议族参数协商的基本信息，如表11。

表 11 SSProtocolProposal 数据结构

| 属性名称 | 数据类型 | 数据说明 | 示例 | 数据备注 |
| --- | --- | --- | --- | --- |
| supported_versions | int32 list | 支持的版本列表 | [1] | 必选 |
| supported_protocols | int32 list | 支持的 SS 协议种类的列表，如 Semi2K | 见表 12 | 必选 |
| field_types | int32 list | 支持的整数环的比特数的列表，如 64、128 | 见表 13 | 必选 |
| trunc_modes | TruncationModeProposal list | 支持的截断方式列表 | 见表 14 | 必选 |
| prg_configs | PrgConfigProposal list | 支持的伪随机数生成器配置列表 | 见表 16 | 必选 |
| shard_serialize_formats | int32 list | 支持的秘密分享分片的序列化格式的 enum 值，如 raw | 见表 18 | 必选 |
| triple_configs | TripleConfigProposal list | 支持的 TTP beaver 服务配置列表 | 见表 19 | 必选 |

- supported_protocols元素的取值和说明如表12所示。

表 12 supported_protocols 元素的值域表

| 取值 | 数据说明 |
|---|---|
| 1 | PROTOCOL_KIND_SEMI2K，Semi2K 协议 |
| 2 | PROTOCOL_KIND_ABY3，ABY3 协议 |

- field_types元素的取值和说明如表13所示。

表 13 field_types 元素的值域表

| 取值 | 数据说明 |
|---|---|
| 2 | FIELD_TYPE_64，整数环 64 |
| 3 | FIELD_TYPE_128，整数环 128 |

- TruncationModeProposal包括截断方式参数协商的基本信息，如表14。

表 14 TruncationModeProposal 数据结构

| 属性名称 | 数据类型 | 数据说明 | 示例 | 数据备注 |
|---|---|---|---|---|
| supported_versions | int32 list | 支持的版本列表 | [1] | 必选 |
| method | int32 | 截断方法，其值域如表 15 所示 | 1 | 必选 |
| compatible_protocols | int32 list | 支持该截断方法的 ss 协议列表，如果留空，表示所有 ss 协议都支持。 | 见表 12 | 必选 |

- method字段的取值和说明如表15所示。

表 15 method 的值域表

| 取值 | 数据说明 |
|---|---|
| 1 | TRUNC_MODE_PROBABILISTIC，概率误差截断 |
| 2 | TRUNC_MODE_PRECISE，高精度截断 |

- PrgConfigProposal包括伪随机数生成器参数协商的基本信息，如表16。

表 16 PrgConfigProposal 数据结构

| 属性名称 | 数据类型 | 数据说明 | 示例 | 数据备注 |
|---|---|---|---|---|
| supported_versions | int32 list | 支持的版本列表 | [1] | 必选 |
| crypto_type | int32 | 加密方式 | 见表 17 | 必选 |

- crypto_type字段的取值和说明如表17所示。

表 17 crypto_type 字段的值域表

| 取值 | 数据说明 |
|---|---|
| 1 | CRYPTO_TYPE_AES128_CTR，AES128_CTR 加密 |
| 2 | CRYPTO_TYPE_SM4_CTR，SM4_CTR 加密 |

- shard_serialize_formats元素的取值和说明如表18所示。

表 18 shard_serialize_formats 元素的值域表

| 取值 | 数据说明 |
|---|---|
| 1 | SHARED_SERIALIZE_FORMAT_RAW |

- TripleConfigProposal包括TTP服务参数协商的基本信息，如表19所示。

表 19 TripleConfigProposal 数据结构

| 属性名称 | 数据类型 | 数据说明 | 示例 | 数据备注 |
|---|---|---|---|---|
| supported_versions | int32 list | 支持的版本列表 | [1] | 必选 |
| sever_version | int32 | TTP 服务的版本号 | 1 | 必选 |

- LrDataIoProposal包括输入输出参数协商的基本信息，如表20。

表 20 LrDataIoProposal 数据结构

| 属性名称 | 数据类型 | 数据说明 | 示例 | 数据备注 |
|---|---|---|---|---|
| supported_versions | int32 list | 支持的版本列表 | [1] | 必选 |
| sample_size | int64 | 样本数量 | 10000 | 必选 |
| feature_num | int32 | 请求方持有的特征数量 | 5 | 必选 |
| has_label | bool | 请求方是否持有标签列 | true | 必选 |

### 7.2 HandshakeResponse 消息

HandshakeResponse消息结构包括算法协商握手响应的基本信息，其数据结构如表21所示。

表 21 HandshakeResponse 数据结构

| 属性名称 | 数据类型 | 数据说明 | 示例 | 数据备注 |
|---|---|---|---|---|
| header | ResponseHeader | 握手请求响应消息头 | 见表 22 | 必选 |
| algo | int32 | 决策下来的算法的 enum 值 | “SS-LR” | 必选 |
| algo_param | google.protobuf.Any | 决策下来的算法详细参数，实际类型随算法而变，LR 算法的实际类型是 LrHyperparamsResult。 | 见表 24 | 必选 |
| ops | int32 list | 复杂算子类型的 enum 值，如 | [1] | 必选 |

| 属性名称 | 数据类型 | 数据说明 | 示例 | 数据备注 |
|---|---|---|---|---|
| | | sigmoid | | |
| op_params | google.protobuf.Any list | 决策下来的复杂算子详细参数，实际类型随复杂算子类型而变，sigmoid 算子的类型是 SigmoidParamsResult。 | 见表 26 | 必选 |
| protocol_families | int32 list | 决策下来的协议族 | [1] | 必选 |
| protocol_family_params | google.protobuf.Any list | 决策下来的协议族详细参数，实际类型随协议族而变，SS 的类型是 SSProtocolResult。 | 见表 27 | 必选 |
| io_param | google.protobuf.Any | 决策下来的 LR 算法输入输出详细参数。实际类型随算法类型而变，LR 算法的类型是 LrDataIoResult。 | 见表 31 | 必选 |

其中：

- ResponseHeader消息结构包括的基本信息如表22所示。

表 22 ResponseHeader 数据结构

| 属性名称 | 数据类型 | 数据说明 | 数据备注 |
|---|---|---|---|
| error_code | int | 握手响应的结果，其值域如表 23 所示。 | 必选 |
| error_msg | string | 用户自定的消息字符串 | 可选 |

- error_code的取值和说明如表23所示。

表 23 error_code 的值域表

| 错误码 | 数据说明 |
|---|---|
| 0 | 成功 |
| 31100000 | GENERIC_ERROR，通用错误 |
| 31100001 | UNEXPECTED_ERROR，状态不符合预期错误 |
| 31100002 | NETWORK_ERROR，网络通信错误 |
| 31100100 | INVALID_REQUEST，非法请求 |
| 31100101 | INVALID_RESOURCE，运行资源不满足 |
| 31100200 | HANDSHAKE_REFUSED，握手拒绝 |
| 31100201 | UNSUPPORTED_VERSION，不支持的版本 |
| 31100202 | UNSUPPORTED_ALGO，不支持的算法 |
| 31100203 | UNSUPPORTED_PARAMS，不支持的算法参数 |

- LrHyperparamsResult包括LR算法参数协商的基本信息，如表24所示。

表 24 LrHyperparamsResult 数据结构

| 属性名称 | 数据类型 | 数据说明 | 示例 | 数据备注 |
|---|---|---|---|---|
| version | int32 | 支持的算法版本 | 1 | 必选 |
| optimizer_name | int32 | 决策下来的迭代优化器类型 | 1 | 必选 |
| optimizer_param | google.protobuf.Any | 优化器参数，实际类型随优化器类型而变，SGD 的类型是 SgdOptimizer | 见表 25 | 必选 |
| num_epoch | int64 | min-batch 梯度下降的 epoch 参数 | 10 | 必选 |
| batch_size | int64 | min-batch 梯度下降的 batch_size 参数 | 1000 | 必选 |
| last_batch_policy | int32 | 决策下来的 mini-batch 梯度下降处理最后一个 batch 的策略 | 1 | 必选 |
| l0_norm | double | 决策下来的 l0 正则项参数，若等于 0 则表示不使用 l0 正则项 | 0 | 必选 |
| l1_norm | double | 决策下来的 l1 正则项参数，若等于 0 则表示不使用 l1 正则项 | 0 | 必选 |
| l2_norm | double | 决策下来的 l2 正则项参数，若等于 0 则表示不使用 l2 正则项 | 0.5 | 必选 |

- SgdOptimizer包括SGD优化器参数协商的基本信息，如表25所示。

表 25 SgdOptimizer 数据结构

| 属性名称 | 数据类型 | 数据说明 | 示例 | 数据备注 |
|---|---|---|---|---|
| learning_rate | double | 学习率 | 0.0001 | 必选 |

- SigmoidParamsResult包括sigmoid算子参数协商的基本信息，如表26。

表 26 SigmoidParamsResult 数据结构

| 属性名称 | 数据类型 | 数据说明 | 示例 | 数据备注 |
|---|---|---|---|---|
| version | int32 | 支持的算法版本 | 1 | 必选 |
| sigmoid_mode | int32 | 决策下来的 sigmoid 逼近公式类型 | 1 | 必选 |

- SSProtocolResult包括SS协议族参数协商的基本信息，如表27。

表 27 SSProtocolResult 数据结构

| 属性名称 | 数据类型 | 数据说明 | 示例 | 数据备注 |
|---|---|---|---|---|
| version | int32 | 版本号 | 1 | 必选 |
| protocol | int32 | 决策下来的 SS 协议种类 | 1 | 必选 |

| 属性名称 | 数据类型 | 数据说明 | 示例 | 数据备注 |
|---|---|---|---|---|
| field_type | int32 | 决策下来的整数环的比特数 | 64 | 必选 |
| trunc_mode | TruncationModeResult | 决策下来的截断方式 | 见表 28 | 必选 |
| prg_config | PrgConfigResult | 决策下来的伪随机数生成器配置 | 见表 29 | 必选 |
| shard_serialize_format | int32 | 决策下来的秘密分享分片的序列化格式 | 1 | 必选 |
| triple_config | TripleConfigProposal | 决策下来的 TTP beaver 服务配置 | 见表 30 | 必选 |
| fxp_fraction_bits | int32 | 定点数的小数位数 | 18 | 必选 |

- TruncationModeResult包括截断方式参数协商的基本信息，如表28所示。

表 28 TruncationModeResult 数据结构

| 属性名称 | 数据类型 | 数据说明 | 示例 | 数据备注 |
|---|---|---|---|---|
| version | int32 | 版本号 | 1 | 必选 |
| method | int32 | 截断方法，其值域如表 15 所示。 | 1 | 必选 |

- PrgConfigResult包括伪随机数生成器参数协商的基本信息，如表29所示。

表 29 PrgConfigResult 数据结构

| 属性名称 | 数据类型 | 数据说明 | 示例 | 数据备注 |
|---|---|---|---|---|
| version | int32 | 版本号 | 1 | 必选 |
| crypto_type | int32 | 加密方式，其值域如表 17 所示 | 1 | 必选 |

- TripleConfigResult包括TTP服务参数协商的基本信息，如表30所示。

表 30 TripleConfigResult 数据结构

| 属性名称 | 数据类型 | 数据说明 | 示例 | 数据备注 |
|---|---|---|---|---|
| version | int32 | 版本号 | 1 | 必选 |
| server_host | string | TTP 服务主机的 ip:port | ip:port | 必选 |
| sever_version | int32 | TTP 服务的版本号 | 1 | 必选 |
| session_id | string | 与 TTP 服务的会话 id | interconnection-session-xxxyyy | 必选 |
| adjust_rank | int32 | 哪一方体用 TTP 服务的 Adjust 接口 | 0 | 必选 |

- LrDataIoResult包括LR算法输入输出参数的基本信息，如表31。

表 31 LrDataIoResult 数据结构

| 属性名称 | 数据类型 | 数据说明 | 示例 | 数据备注 |
|---|---|---|---|---|
| version | int32 | 版本号 | 1 | 必选 |
| sample_size | int64 | 样本数量 | 10000 | 必选 |
| feature_nums | int32 list | 各方持有的特征数量，例如两个参与方的情况下，若 rank 0 有 3 个特征，rank 1 有 4 个特征，则 feature_nums 等于 [3, 4] | [3,4] | 必选 |
| label_rank | int32 | 哪一方持有标签 | 1 | 必选 |

# 8 传输层实现参考

## 8.1 通信框架

异构隐私计算技术平台间进行互联互通时，通信框架需能满足兼容性、通用性以及安全性的要求，如表32表32所示。

表 32 通信框架范围

| RPC框架 | GRPC |
|---|---|
| 编码方式 | ProtoBuf |

## 8.2 初始化通信协议

初始化通信协议在 SS-LR 任务开始前执行一次。

每个参与者向其它参与者通知自己的存在性，即向他人发送 connect_{self_rank}：

For i in 0..word_size [注1]:

```
    if i == self_rank:
        continue
    P2P send to rank i: {key: connect_{self_rank}, value: ""}
```

每个参与者检查他人的存在性，即依次检查 connect_{rank} 消息已经收到：

```
For i in 0..word_size:
    if i == self_rank:
        continue
    P2P receive on key connect_{i}
```

注 1：word_size 表示参与者数量

## 8.3 Protobuf 消息

不同隐私计算的参与者之间使用 Protobuf 协议传递信息。

```
service ReceiverService
{
    rpc Push(PushRequest) returns (PushResponse);
}
```

### 8.3.1 PushRequest 消息结构

PushRequest包括传输的基本信息，其数据结构如表33所示。

表 33 PushRequest 数据结构

| 属性名称 | 数据类型 | 数据说明 | 示例 | 数据备注 |
|---|---|---|---|---|
| sender_rank | uint64 | 发送者的编号，如 0 指 rank | 0 | 必选 |
| key | string | 消息唯一 ID，生成规则见 8.4 节 | “root:P2P-0:0->1” | 必选 |
| value | bytes | 消息体，ECDH-PSI 中的 | 需要传输 | 必选 |

| 属性名称 | 数据类型 | 数据说明 | 示例 | 数据备注 |
|---|---|---|---|---|
| | | protobuf 序列化成二进制 string，然后把整个 string 放到 value 中 | 的实际信息 | |
| trans_type | TransType | 传输模式，如全量传输、分块传输 | 见表 34 | 必选 |
| chunk_info | ChunkInfo | 消息大小 | 见表 35 | 必选 |

其中：

- TransType 的值域如表 34 所示。

表 34 TransType 的值域表

| 数值 | 数据说明 |
|---|---|
| MONO | 全量传送模式 |
| CHUNKED | 分块传送模式 |

- ChunkInfo 的值域如表 35 所示。

表 35 ChunkInfo 的数据结构

| 属性名称 | 数据类型 | 数据说明 | | 数据备注 |
|---|---|---|---|---|
| message_length | uint64 | 数据总大小，单位是字节 Byte | 1048576 | 必选 |
| chunk_offset | uint64 | 当前分块的偏移量 | 0 | 必选 |

### 8.3.2 PushResponse 消息结构

PushResponse包括传输的基本信息，其数据结构如表36所示。

表 36 PushResponse 消息的数据结构

| 属性名称 | 数据类型 | 数据说明 | 数据备注 |
|---|---|---|---|
| header | ResponseHeader | 返回消息，如表 22 所示 | 必选 |

## 8.4 通信模式

### 8.4.1 信道

信道是一个逻辑概念，用于区分通信的上下文。每一个信道有一个全局唯一名称，命名规则为：\w+，即信号名称由字母、数字、下划线组成。信道的名字由通信组双方约定，在初始化阶段由用户传入。

信道唯一的作用就是影响 message key 的生成，信道名称会作为 message key 一部分，因此，不同信道中的消息一定不会有相同的 key，因此不同信道的消息在逻辑上不会混淆。

当上层算法需要多个信道时，第一个信道称为主信道，其它信道称为子信道。子信道的命名规则为：主信道名称-子信道编号。

举例：假设主信道名称为 root，则 0 号子信道名称为 root-0，1 号子信道名称为 root-1，以此类推。

### 8.4.2 P2P 通信

P2P 通信允许在任意两个参与者之间发送信息。P2P 通信 key 的命名规则为：{信道名称}:P2P-{计数器}:{发送者 RANK}->{接收者 RANK}，其中每一个信道、每一对 <sender，receiver> 都有一个独立的计数器。

举例，假设信道名称为 root，以下消息依次发送：

Rank 0 → 1 发送消息，key 为：root:P2P-0:0->1

Rank 1 → 0 发送消息，key 为：root:P2P-0:1->0

Rank 0 → 2 发送消息，key 为：root:P2P-0:0->2

Rank 0 → 1 发送消息，key 为：root:P2P-1:0->1

### 8.4.3 Allgather 通信

Allgather通信的作用是每个参与方将自己的密态分片发送给其他所有参与方，这样每个参与方都汇集了数据的所有分片，从而每个参与方都可以得到数据的明文。

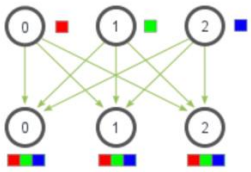


图 2 Allgather 通信

Allgather通信key的命名规则为{信道名称}:{计数器}:ALLGATHER,其中每一个信道都有一个独立的计数器。

举例，假设 rank 0 和 rank 1 要 Allgather 数据，则协议流程如下：

时间点 1：

RANK 0 构建 PushRequest 发送给 RANK 1，其中 key 为 root:1:ALLGATHER

RANK 1 构建 PushRequest 发送给 RANK 0，其中 key 为 root:1:ALLGATHER

RANK 0 接收 来自 RANK 1 的 key 为 root:1:ALLGATHER 的数据

RANK 1 接收 来自 RANK 0 的 key 为 root:1:ALLGATHER 的数据

# 附 录 A
（资料性）
## 样本数据集初始化秘密分享示例

若样本总数量等于 5，样本特征数量等于数组[2, 3]，参与方 A（rank=0）拥有 2 个维度的特征和 5 个样本的标签，参与方 B（rank=1）拥有 3 个维度的特征。

双方对样本数据集进行秘密分享。参与方 A 初始化秘密分享的结果如图 A.1 所示。

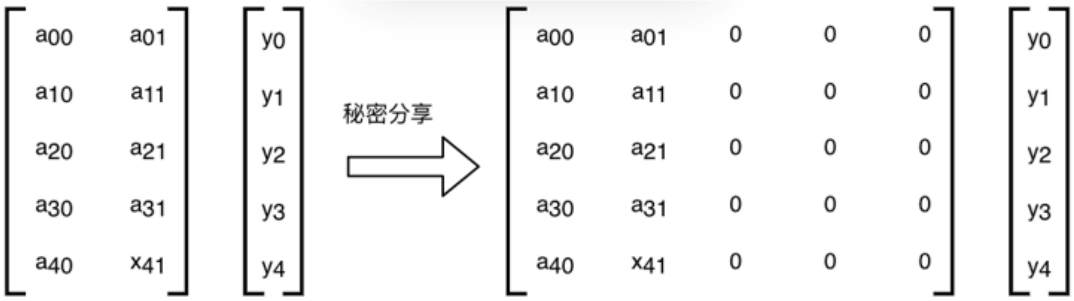


图 A.1 参与方 A 样本数据集秘密分享示例

参与方 B 初始化秘密分享的结果如图 A.2 所示。

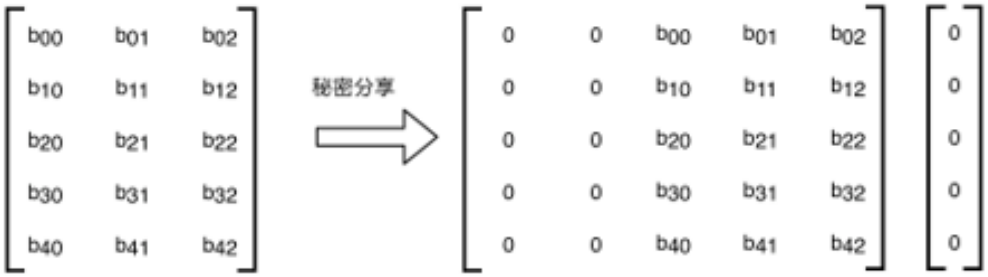


图 A.2 参与方 B 样本数据集秘密分享示例

# 附 录 B
（资料性）
可信第三方 Beaver 服务

Beaver 服务接口有三个，分别是 CreateSession、DeleteSession、AdustDot。

## B.1 创建会话

CreateSession 接口在 Beaver 服务端创建一个会话，并保存参与方上传的随机数种子。程序启动后，各参与方向 Beaver 服务发送创建会话的请求，并在请求中将各自的随机数种子发送给 Beaver 服务，如图 B.1 图所示。

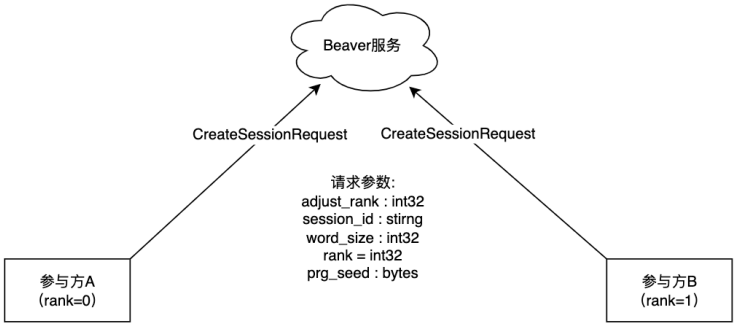


图 B.1 各参与方与 Beaver 服务建立会话

CreateSession 请求参数中：

- adjust_rank 表示哪个参与方获取$adjust$值；
- session_id 是当前会话编号；
- word_size 是参与方的数目；
- rank 是参与方的编号；
- prg_seed 是随机数种子字段。该字段是一个字节序列，数据结构如表 B.1 37B.1 所示。

表 B.1 37 Int128Proto 数据结构

| 属性名称 | 数据类型 | 数据说明 | 示例 | 数据备注 |
|---|---|---|---|---|
| hi | int64 | 随机数种子的高 64 位 | 随机值 | 必选 |
| lo | uint64 | 随机数种子的低 64 位 | 随机值 | 必选 |

## B.2 获取$adjust$值

如果建立会话时确定的 adjust_rank 等于 0，则参与方 A 调用 Beaver 服务的 AdjustDot 接口获得 $adjust$ 值，然后将 $adjust$ 加在 $C_0$ 上，具体过程如图 B.2 所示。

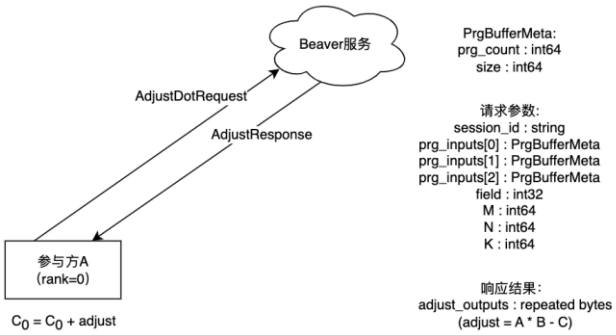


图 B.2 一方从 Beaver 服务获取矩阵乘法三元组的调整值（$adjust = A \cdot B - C$）

AdjustDotRequest 参数中：

- session_id 表示当前会话编号；
- prg_inputs[0]表示生成随机数矩阵$A_i$的需要用到的相关输入；
- prg_inputs[1]表示生成随机数矩阵$B_i$的需要用到的相关输入；
- prg_inputs[2]表示生成随机数矩阵$C_i$的需要用到的相关输入；
- field 是整数环的比特数；
- M 等于随机数矩阵$A_i$和$C_i$的行数；
- N 等于随机数矩阵$B_i$和$C_i$的列数；
- K 等于随机数矩阵$A_i$的列数和$B_i$的行数；
- prg_count 是生成随机数矩阵需要用到的计数器值；
- size 是随机数矩阵的字节数。

AdjustResponse 参数包含：

- adjust_outputs 是矩阵乘法三元组的调整值。

AdustDot 接口根据请求参数中的三个当前计数器值字段和比特序列长度字段，结合会话中保存的两个随机数种子，计算出请求方本地伪随机数的调整值 $adjust$。每个参与方在本地维护一个私有的随机数种子和一个公共的当前计数器值。当前计数器值的初始值为 0，在 LR 算法执行时随着伪随机数生成过程不断累加。参与双方各自维护的当前计数器值应始终保持一致。

### B.3 删除会话

程序结束时，任意一方（或双方同时）调用 DeleteSession 接口删除会话，如图 B.3 所示。

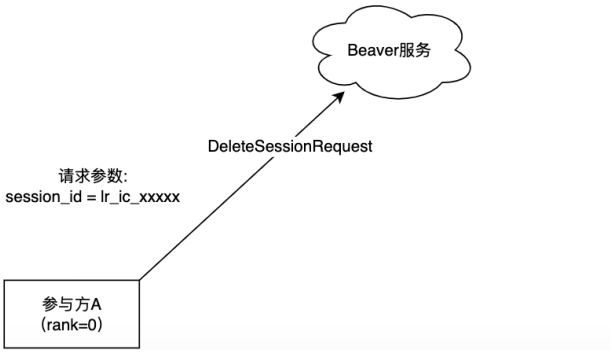


图 B.3 程序结束时请求 Beaver 服务删除会话

DeleteSessionRequest 参数包含：

- session_id 表示当前会话编号。

# 参 考 文 献

[1] NIST SP 800-57 Part1 Rev. 5 Recommendation for Key Management: Part 1 – General

[2] GB/T 25069-2022 信息安全技术 术语

[3] GM/T 32905 - 2016 信息安全技术 SM3 密码杂凑算法

[4] GB/T 32918-2017 信息安全技术 SM2 椭圆曲线公钥密码算法

[5] NIST SP 800-90Ar1: Recommendation for Random Number Generation Using Deterministic Random Bit Generators

[6] GM/T 0105-2021 软件随机数发生器设计指南