**一次多项式安全多方计算Java API接口说明**

# Java包结构

抽象类com.viewsources.crypto.mpc.LinearMultipartyComputation抽象定义了一次多项式安全多方计算的接口并实现了公共代码。其实现子类LinearMultipartyComputationInteger和LinearMultipartyComputationBigInteger分别实现了Integer和BigInteger域下的安全多方计算特定代码。

上述安全多方计算依赖以下算法代码：

1. com.viewsources.crypto.secretshare.ShamirSecretSharing：Shamir的秘密分享算法；
2. com.viewsources.crypto.poly.Polynomial：一元多次多项式计算；
3. com.viewsources.crypto.poly.LinearEquationSystem：多元一次方程组。

# Preliminary

1. *n*：参与安全多方计算的实体数；
2. id：每个实体所分配到一个唯一的ID号，数值从0 ~ *n*-1；
3. *f*(*x*1, *x*2, …, *xn*) = *c*1*x*1 + *c*2*x*2 + … + *cnxn*：所需安全多方计算的一次多项式。其中*c*1, *c*2, …, *cn*为公共常数，*x*1, *x*2, …, *xn*是*n*个参与实体各自的秘密信息；
4. *t*：threshold，semi-honest恶意实体数，需小于*n*/2。所以semi-honest是指实体会正确执行算法和协议，但会勾结起来尝试恢复其他实体的秘密信息。
5. *max* = max(*x*1, *x*2, …, *xn*)，秘密信息的最大值。

# 创建安全多方计算对象

每个实体通过以下方案创建安全多方计算对象(假定创建的是32位Integer MPC对象)：

LinearMultipartyComputation mpc = new LinearMultipartyComputationInteger(id, participantCount, threshold);

参数说明如下：

1. id：实体分配到的ID；
2. participantCount：参与计算的实体总数；
3. threshold：恶意实体数(需小于*n*/2)。

# 准备阶段

计算发起者根据多项式*f*和秘密信息的最大值*max*，确定Shamir秘密分享方案中的素数模*p*，选择一组随机数；并发送给其他参与实体，代码如下：

mpc.initializeWithMaxValue(coefficients, max);

Object modulus = mpc.getModulus();

Object[] auxiliary = mpc.generateInputAuxiliary();

其中：

1. 参数coefficients：多项式*f*的系数数组*c*1, *c*2, …, *cn*；
2. 参数max：秘密信息的最大值*max*；
3. 返回值modulus：hamir秘密分享方案中的素数模*p*；
4. 返回值auxiliary：随机数。

将modulus和auxiliary[*i*]发送给ID为*i*的实体。

其他实体接收到modulus和auxiliary[*i*]后，通过下述代码进行初始化：

mpc.initializeWithModulus(coefficients, modulus);

# 输入阶段

每个参与实体计算输入值，代码如下：

Object[] inputs = mpc.generateInputs(secret, auxiliary);

其中：

1. 参数secret：参与实体的秘密信息*xi*；
2. 参数auxiliary：随机数；由发起者生成，发送给其他实体；
3. 返回值：输入值数组。

将inputs[*i*]发送给ID为*i*的实体。

实体在收到其他实体发送来的输入值input后，调用以下代码加入：

mpc.addReceivedInput(from, input);

其中：

1. 参数from：输入值发送者的ID；
2. 参数input：接收到的输入值。

实体可以通过以下代码判断是否接收到了所有的输入值：

boolean ready = mpc.hasAllInputReceived();

# 输出阶段

每个参与实体计算一个输出值：代码如下：

Object output = mpc.generateOutput();

将输出值output发送给其他所有实体。

实体在收到其他实体发送来的输出值output后，调用以下代码加入：

mpc.addReceivedOutput(from, output);

其中：

1. 参数from：输入值发送者的ID；
2. 参数output：接收到的输出值。

实体可以通过以下代码判断是否接收到了足够的输出值以进行最后的计算：

boolean ready = mpc.isReadyForCompute();

# 计算阶段

参与实体计算出多项式*f*的值，代码如下：

Object result = mpc.compute();