**安全内积协议**

# 功能介绍：

Input：

Alice owns X =（x1，x2 , …, xn）;

Bob owns Y = ( y1, y2, …, yn);

Output

Alice outputs X · Y = ( x1,x2,...,xn ) · ( y1,y2,...yn ) = x1y1 + x2y2 + ... xnyn

Bob outputs empty;

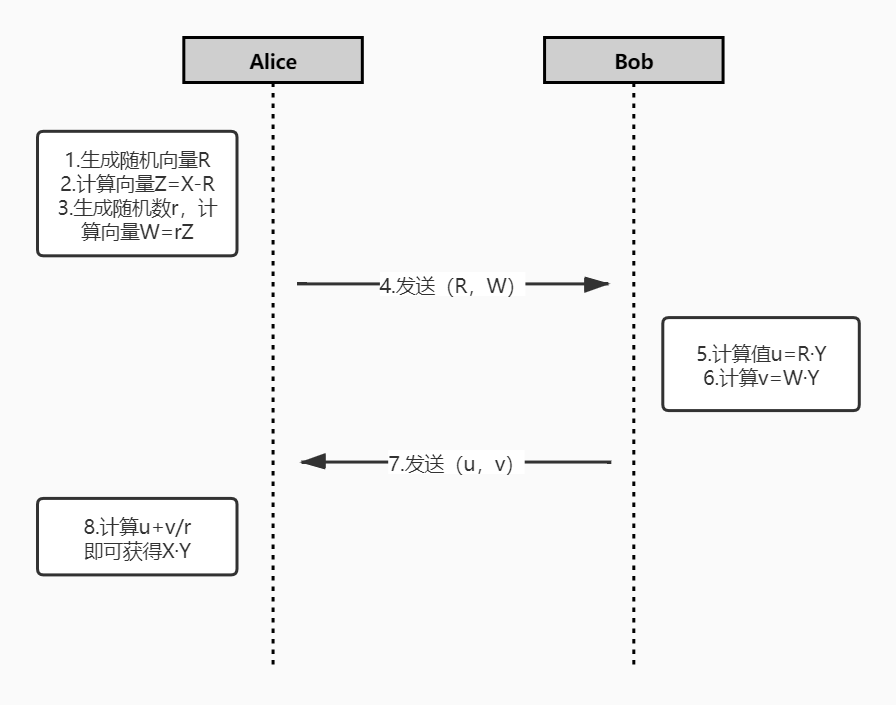
# 说明

此协议Alice最终获得内积结果

# 具体协议内容：

1. Alice选择随机数向量 R = （r1, r2, …., rn）, ri ∈Fp ，计算Z = X – R =（z1, z2, …., zn）,zi = xi - ri∈Fp； 在选择一个随机数r∈Fp，计算W = rZ = （rz1, rz2, …, rzn）,wi = rzi∈Fp；发送（R，W）给Bob。
2. Bob计算 u = R·Y = （r1y1 + r2y2 + … + rnyn）, u∈Fp，计算v = W·Y = (w1y1 + w2y2 + … + wnyn), v∈Fp ，发送（u，v）给Alice。
3. Alice计算u+ v/r就可以获得X·Y。

# 流程定义：

程序流程图如下：

交互流程图

流程描述：

1. 初始化生成掩码向量R
2. 计算加了掩码向量Z
3. 生成随机数r，计算向量W=rZ
4. 利用Sockets发送（R，W）
5. Bob计算u = R·Y
6. Bob计算v = W·Y
7. 利用Sckets发送值（u，v）
8. Alice计算u+v/r即可获得结果

# 接口定义：

void AliceInitInnerProduct（uint32\_t val\_X[ ]，uint32\_t mask\_R[ ], uint32\_t mask\_Z[ ] , uint32\_t blinding\_r , uint32\_t mask\_W[ ]）{

}

void BobInnerProduct（uint32\_t mask\_R[ ] , uint32\_t mask\_W[ ], uint32\_t bob\_u, uint32\_t bob\_v）{

}

int AliceInnerProduct(uint32\_t bob\_u, uint32\_t bob\_v, uint32\_t blinding\_r){

return InnerProductResult;

}

//csocket接口现在没看 不知道要不要写 怎么写

//#if DJN\_CHECKMT

    std::cout << "Checking MT validity with values from other party:" << std::endl;

    mpz\_t ai, bi, ci, ai1, bi1, ci1, ta, tb;

    mpz\_inits(ai, bi, ci, ai1, bi1, ci1, ta, tb, NULL);

    chan->send(bA, numMTs \* shareBytes);

    chan->blocking\_receive(bA, numMTs \* shareBytes);

    chan->send(bB, numMTs \* shareBytes);

    chan->blocking\_receive(bB, numMTs \* shareBytes);

    chan->send(bC, numMTs \* shareBytes);

    chan->blocking\_receive(bC, numMTs \* shareBytes);

    for (uint32\_t i = 0; i < numMTs; i++) {

        mpz\_import(ai, 1, 1, shareBytes, 0, 0, bA + i \* shareBytes);

        mpz\_import(bi, 1, 1, shareBytes, 0, 0, bB + i \* shareBytes);

        mpz\_import(ci, 1, 1, shareBytes, 0, 0, bC + i \* shareBytes);

        mpz\_import(ai1, 1, 1, shareBytes, 0, 0, bA1 + i \* shareBytes);

        mpz\_import(bi1, 1, 1, shareBytes, 0, 0, bB1 + i \* shareBytes);

        mpz\_import(ci1, 1, 1, shareBytes, 0, 0, bC1 + i \* shareBytes);

        mpz\_add(ta, ai, ai1);

        mpz\_add(tb, bi, bi1);

        mpz\_mul(ta, ta, tb);

        mpz\_add(tb, ci, ci1);

        mpz\_mod\_2exp(ta, ta, m\_nShareBitLength);

        mpz\_mod\_2exp(tb, tb, m\_nShareBitLength);

        if (mpz\_cmp(ta, tb) == 0) {

            std::cout << "MT is fine - i:" << i << "| " << ai << " " << bi << " " << ci << " . " << ai1 << " " << bi1 << " " << ci1 << std::endl;

        } else {

            std::cout << "Error in MT - i:" << i << "| " << ai << " " << bi << " " << ci << " . " << ai1 << " " << bi1 << " " << ci1 << std::endl;

        }

        //std::cout << (mpz\_cmp(c1[i], a1[i]) == 0 ? "MT is fine." : "Error in MT!") << std::endl;

    }

    mpz\_clears(ai, bi, ci, ai1, bi1, ci1, ta, tb, NULL);

//#endif