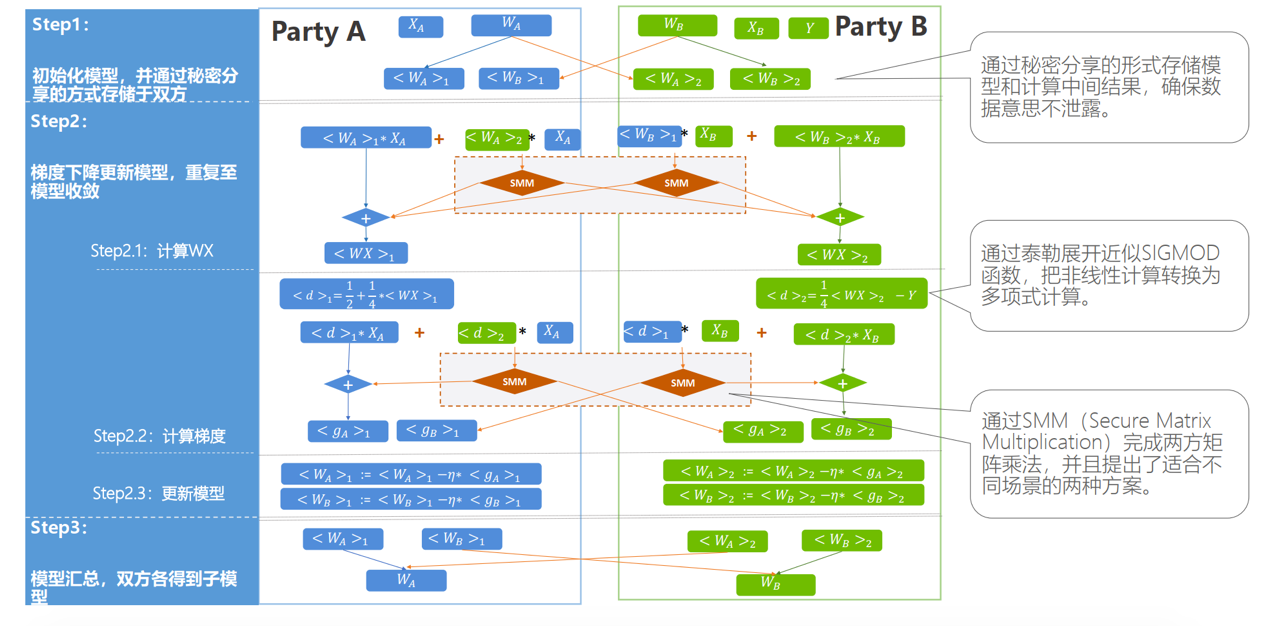
# 蚂蚁多方安全计算的评估总结

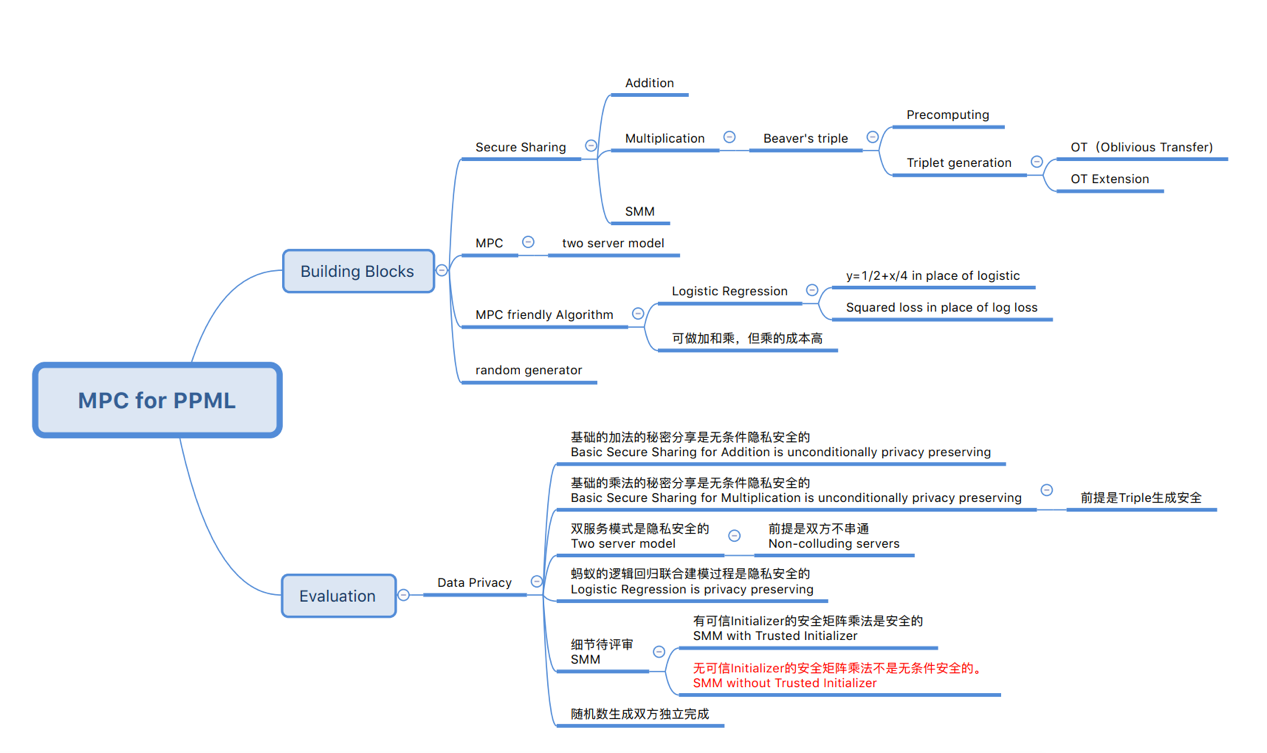
## 概述

蚂蚁所提出的联合建模的安全框架为多方安全计算（MPC, secure MultiParty Computation）, 具体来说采用了两服务者模式（Two Server Model）。即所有参与计算的客户（虽然平安和蚂蚁是两方，但不限于两方）使用两个服务（此处一个位于蚂蚁，一个位于平安）来完成安全计算，只要这两个服务器不存在串通行为，计算就是安全的。除两服务者模式以外，还有2-4模式等。



图中XB, Y, Wb，gb(行方数据)自始至终都没有出域。图中<WB>1\*XB, <d>1\*XB看上去有些误导,实际上<WB>1，<d>1等都不出域，双方在一起做安全矩阵乘法(SMM), 而安全矩阵乘法计算的过程中双方交互的都是随机数。

## 蚂蚁多方安全计算的思路剖析



蚂蚁多方安全计算的基本思路是这样，对于银行的数据合作，首先选择了个安全优先的框架即多方安全计算，确认了并非沿袭联邦学习的架构。为了避免这个框架对于机器学习这种计算密集型效率很低这个问题，他们从模型、场景、优化各个环节加以调整，寻找可以落地的Case。从模型上, 用了一个接近线性的link function替换了logit（logistic function泰勒展开最高只取到一次项），由于迭代过程中为了安全，不泄漏残差，d始终没有合到一起，终止条件改成固定迭代次数等等。

这个架构我们在另一个项目评估过，安全性是数学上可证明的，没有梯度反推原始数据的那种风险。基本的密码学组建都是都是坚实的，蚂蚁所做的工作主要都是些技术选择使整体的计算复杂度可以接受。

总体的安全性基于随机数的安全上, 这个点是有一点风险的，这个跟他们探讨过，他们的回应是，随机数生成可各自实现，这个可以在落地的时候再考虑。

整个过程蚂蚁的所得是WA, 这是一个知识，他们有可能可以用这个来改进自己的模型，比如集成学习之类的。另一个我们可能需要防范的点是推理的部分，蚂蚁方可能有机会对行方的调用进行分析，这个我们可以故意掺些沙子，这个细节可以晚些再说。

## 安全组件介绍

### 秘密分享

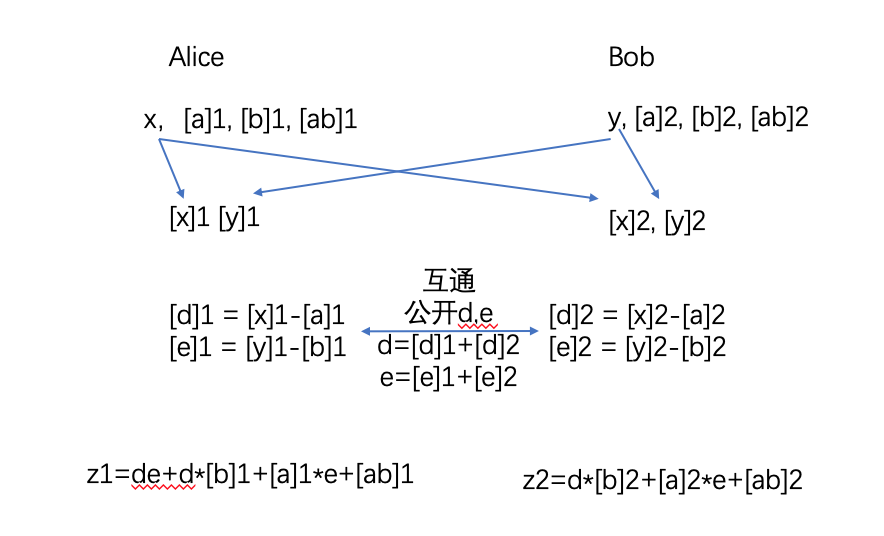
秘密分享算法主要技术依赖于Shamir等人在1979年所提出的秘密分享技术。秘密分享技术是多方安全计算领域中的一项核心技术，被广泛应用于隐私保护机器学习算法中。其基本思想是把一个秘密切分成n份并发送给n个人，每个人保存了整个秘密中的一份share。这些参与者中的一个或者几个人无法窥探到秘密，而当真正要重构这个秘密时，需要拿到share的这n个人共同参与，这叫做 (n, n)秘密分享。在蚂蚁的场景下，为两方的秘密分享。秘密分享的加法由参与方各自完成，非常简单； 而秘密分享的乘法技术则依赖于Beaver在1991年所提出的Beaver’s Triple技术。

下图是秘密分享（X=X1+X2）的一个示意图。加法的共享基于此，比较简单，直接把X,Y各自秘密分享了，再把被分享的部分在两方加起来就可以了: X+Y=(X1+Y1)+(X2+Y2)。



## 乘法分享三元组（Beaver’s Triple）

乘法分享的目标如下，在双方无法得到另一方的信息的前提下完成乘法运算，结果分享在两方。即xy=z1 + z2. 这个问题解法思路如下：将x, y各自独立盲化，（盲化的方法是将其加上一个随机的偏移量），求其积，将xy由盲化后的积复原。xy = ( (x-a) + a)\* ( (y-b)+b)=(d+a)(e+b)=de+db+ae+ab. 在这个式子里d, e（盲化后的x，y）是公开的可视为常量，ab由于是不公开的，也需要被秘密分享，无法各自运算。因此此式中的a, b, ab都要被秘密分享，他们就构成所谓的三元组。实际计算时de作为一个常量，可以只在一方计算。即一方计算： z1= de+d\*[b]1+[a]1\*e +[ab]1, 另一方计算 z2= d\*[b]2+[a]2\*e+[ab]2, 两方合在一起就可以得回 xy。注意到此处d, e为盲化后的共享值， [a]1 [a]2, [b]1, [b]2, [ab]1, [ab]2均为随机数，因此只要三元组是无条件安全的，乘法分享就是无条件安全的。



### 安全矩阵乘法（SMM）

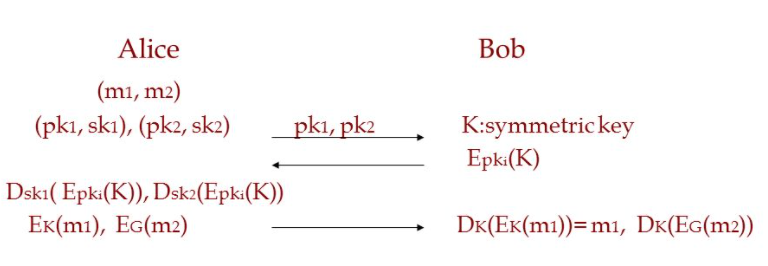
安全矩阵乘法与乘法分享的原理相同。XY = （（X-U）+U）\*（（Y-V）+V）=D\*E+D\*V+U\*E+UV，三元组为U, V, UV。其中D、E为公开的盲化矩阵，U、V、UV为秘密分享的三元组。另外，向量内积从本质上就是矩阵乘法, 所以是SMM的一个子问题。

### 三元组的生成

三元组的生成有多种方法，常用的有通过加性同态加密的方法和通过不经意传输扩展（OT Extension）的方法，其中OT Extension方法的效率较高。

### 不经意传输（OT, Oblivious Transfer）与OT Extension

不经意传输是一种安全计算协议，常在混淆电路的隐私计算中使用。他的核心思想是Bob想知道Alice的两个消息中的一个，比如说m1和m2中的m1但是他不想让她知道自己想要的是1，但Alice同时也要保证Bob只能得到其中一个比如m1，如果拿到了m1，m2他就拿不到了。不经意传输有多种实现方法，其中一种方法如下。



D是解密，E是加密。其中pki i=1或2, 图示i=1 那么Dsk1(Epki(K))等于K，G=Dsk2(Epki(K))，G是一串随机乱码。

一个不经意传输(OT)单元本身只能完成安全计算中一个原子逻辑，却是安全计算的基础。OT已被证明不能以对称密码实现，而只能以非对称密码实现，因而成本极高。学术界一直在研究低成本的协议, 从而使OT能够落地。不经意传输扩展（OT Extension）就是其中的一个方向。OT Extension受类似SSL的协议的启发，既然非对称密码的成本很高，就以一个协议结合非对称密码和对称密码，其中的对称密码就是会话密钥。

OT Extension扩展的方向主要有两个，一个是加长OT的长度，上述例子中，长度为一个bit，不妨称之为纵向。另一个方向是构建更多的OT，不妨称之为横向。横向扩展基于一些基OT，利用对称密码产生更多的OT。OT Extension较为复杂，这里不详细介绍了。