# 安全多方计算与同态加密及其在业界的应用

洪澄

蚂蚁集团 ANT GROUP

蚂蚁技术研究院 Ant Research

## 引子

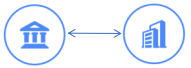
- A公司想知道自己的用户与B公司用户有多少重合度
  - 可以针对重合的这部分用户开展营销活动
  - 但是双方都不想将自己的用户信息发给对方
- 直接哈希手机号求交集行不行?
  - A -> B : { Hash(a) }
  - B local compute { Hash(b) } and compare

## 更多的“密态计算”场景

- A博士想把自己的论文送到TB网站查重一下
  - 但是不想泄露自己的论文具体内容
- B博士想在某网站上查询某个药的使用方法
  - 但是不想泄露自己在关注这个药

## “密态计算” in Ant Group

- 需求背景: 跨机构的数据合作
  - 反洗钱；客户营销;



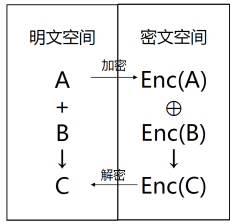
- 外部
  - E.g. 蚂蚁 vs 江苏银行, 蚂蚁 vs 浦发银行 ...
- 内部
  - 不同子公司之间的数据合作
  - E.g. 支付宝 vs 蚂蚁链
- 服务
  - 银行A和银行B通过蚂蚁的密态计算产品进行合作

# Outline

- （全）同态加密 FHE
- 安全多方计算MPC
- 我们的工作简介

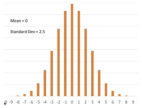
# 同态加密Homomorphic Encryption

- 一种特殊的加密算法
- 密文不仅不泄露明文的信息，还支持互操作
  - E.g. 密文空间的 ⊕ 对应明文加法
  - 密文空间的 ⊙ 对应明文乘法
- 能支持无限次加法和乘法，则称为Fully HE




# BFV（toy example）-- 加密

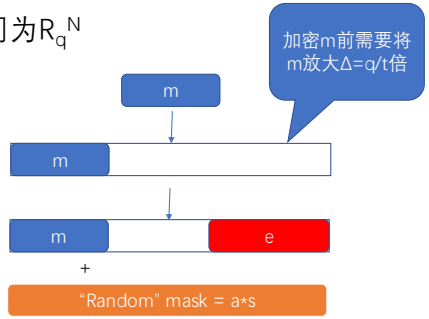
- 记系数mod q的mod $X^N+1$多项式空间为$R_q^N$
- 明文$m \in R_t^N$ //t<q，密钥$s \in R_q^N$
- 则密文为$(m+a*s+e, a)$ //$a \in R_q^N$



e一般很小，如标准差=3.2的高斯分布

安全性：RLWE假设

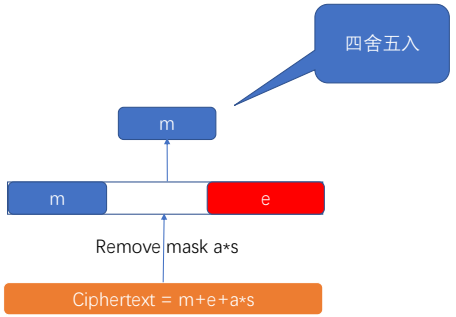
- 已知 (a*s , a)是不安全的：可以反解出s
- 但是加上一些小的e之后，(a*s+e, a)就变成安全的：反解不出s




# BFV（toy example）-- 解密

解密：

- Enc(m) = (m+e+a*s , a) = (A,B)
- A-B*s
  =m+e+a*s-a*s
  = m+e ≈ m

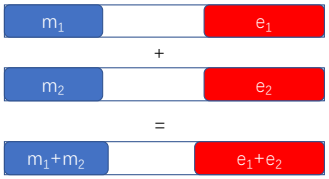


## 错误的同态加密打开方式

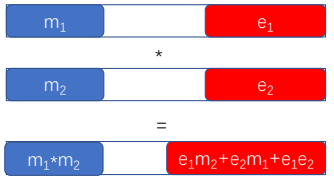
- Affine encryption，具有加法同态性 （腾讯）
  - 密钥生成：取大整数n , a , b ； 公开n；a , b 私密保存。
  - 加密m：Enc(m) = (am+b mod n , 1)，记为(E , k)
  - 密文加法：Add( (E1,k1), (E2,K2) ) = (E1+E2, k1+k2)
  - 解密：Dec(E, k) = $a^{-1}$ *(E – k*b) mod n
- 不安全
  - Enc(m+1) – Enc(m) 就可以破解出a了
  - Enc(0)直接泄露了b
- 加密算法一定是可以正向的论证安全性
  - 反向思路要不得： “我设计一个算法，没人能破解就是安全的”
    - 那只是因为能破解的人没站出来

## BFV（toy example） -- 同态性

满足加法同态性：

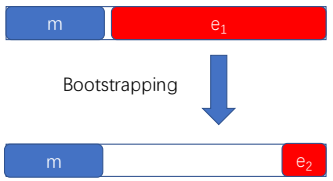



满足乘法同态性：




## 自举Bootstrapping

- 进行多次乘法操作后，噪声导致无法进一步计算
  - 再计算就出错了
- 自举可以“清除”噪声
  - 迭代自举即可实现无限计算
  - 没有自举不能叫FHE




## FHE的编码方式

- BFV的明文是mod $X^N+1$的多项式空间
  - 但是实际应用中，需要处理的是数字而不是多项式
  - 需要一种从数字>多项式的编码机制
- 而且这种编码机制也得是同态的

# 方案A-系数编码

- 例：甲的基因测序是$\{a_0,a_1,\cdots a_{n-1}\}$，乙的基因测序是$\{b_0,b_1,\cdots b_{n-1}\}$
  - 双方希望计算彼此的基因相似度，但不泄露基因明文
- Naïve方案：甲使用n个多项式编码自己的基因，然后加密
  - $\{a_0,a_1,\cdots a_{n-1}\}$ -> $a_0 \bmod X^N+1$, $a_1 \bmod X^N+1$, ⋯, $a_{n-1} \bmod X^N+1$
  - 乙对这些密文和自己的明文，执行加密计算 $\Sigma a_i b_i$，返回给甲
  - 甲解密得到相似度。
- 总共花费是n次加密，n次密文-明文乘，n-1次密文加法，1次解密，通信量n+1个密文。
  - Can we do better?

# 方案A-系数编码 （续）

- 改进方案：甲使用一个多项式编码自己的基因，然后加密

$$P_a = a_0 + a_1X + a_2X^2 + \ldots + a_nX^n,$$

- 乙也使用一个多项式编码自己的基因：

$$P_b = b_0 - b_1X^{N-1} - b_2X^{N-2} - \cdots - b_nX^{N-n}$$

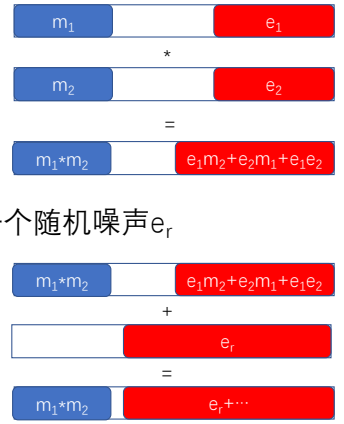
  - 然后对$P_a$的密文和自己的明文$P_b$，执行加密计算 $P_aP_b$，返回给甲
- 甲解密，内积$\Sigma a_i b_i$即在结果$P_aP_b$的常数项上。
  - 原因：$a_1X * (-b_1X^{N-1}) = -a_1b_1X^N = a_1b_1 \bmod X^N+1$
- 总共花费是1次加解密，1次密文-明文乘，通信量2个密文
  - Is it OK ?

- 安全问题：

$$P_a = a_0 + a_1X + a_2X^2 + \ldots + a_nX^n, \qquad P_b = b_0 - b_1X^{N-1} - b_2X^{N-2} - \cdots - b_nX^{N-n}$$

- $P_aP_b$的常数项是$\Sigma a_i b_i$
  - 但是甲解密还能得到$P_aP_b$的其他次项
  - 这些是预期结果之外的额外信息
- 解决方案：乙不是直接发回$Enc(P_aP_b)$，而是发回$Enc(P_aP_b)+P_r$
  - $P_r$是一个常数项=0，其他项是随机数mod t的多项式
  - Is it OK ?

- 安全问题：
- $P_aP_b$的常数项明文是$\Sigma a_i b_i$
  - 但是甲解密得到的不仅是明文，还有噪声
  - 噪声是预期结果之外的额外信息
- 解决方案：乙需要在常数项上额外加上一个随机噪声$e_r$
  - $e_r$需要远大于预期的噪声,且不侵入明文
  - "noise flooding"
  - Is it OK ?
  - Yes!

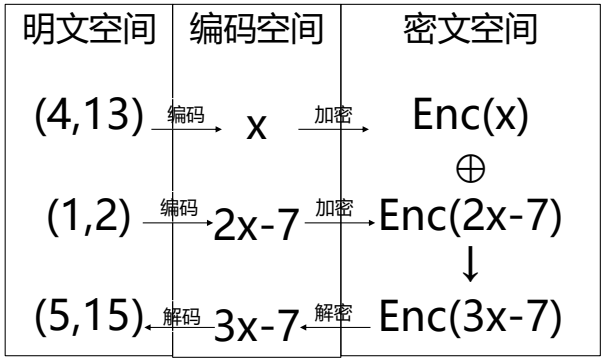


# 方案B-SIMD编码 （Packing）

- 如何将需要计算的数据编码为多项式?
  - $x^n+1$可以拆分为n个多项式的积：$x^n+1=(x+a_1)(x+a_2)....(x+a_n)$
    - 例：设t=17,n=2, 则$x^2+1=(x-4)(x-13)$ // $x^2-17x+52$ mod 17
  - f(x) mod ($x^n+1$) 可以表示n个整数：$x_i$ = f(x) mod $(x+a_i)$
    - 例：x mod ($x^2+1$) 可以表示 x mod (x-4) 和 x mod (x-13)，即 **x mod ($x^2+1$) “pack”了4和13**
- 给定n个整数，可以通过中国剩余定理找到对应的f(x)来编码它们
  - 例：**2x-7 “pack”了1和2**： // 2x-7 mod (x-4) = 1 , 2x-7 mod (x-13) = 2 mod 17
- Packing依然保持 (mod t上的) 同态性
  - 加法: **x+(2x-7) packs 5 and 15**： // 3x-7 mod (x-4) = 5 , 3x-7 mod (x-13) = 15 mod 17
  - 乘法: **x*(2x-7) packs 4 and 9**： // $2x^2-7x$ mod ($x^2+1$) = -7x-2； -7x-2 mod (x-4) = 4，-7x-2 mod (x-13) = 9 mod 17
- SIMD: 一次多项式运算完成了n次整数运算

# 方案B-SIMD编码 （续）

- 一次密文加/乘可以完成N次明文加/乘，故名SIMD
  - Single Instruction Multiple Data

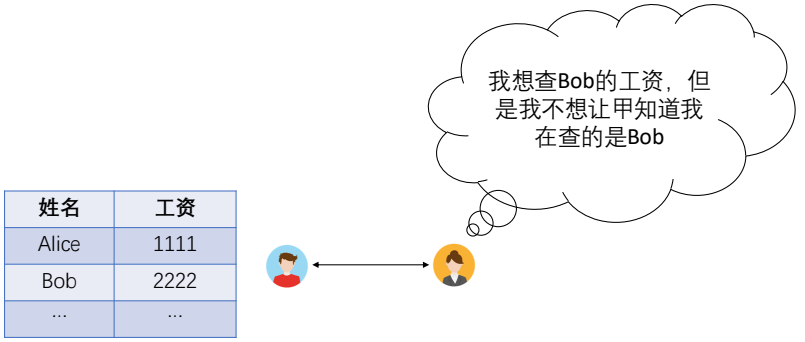



- 例：商家购买云上的数据库保存自己的订单数据
  - 但又害怕云服务商看到自己的营业金额，因此使用同态加密

| 订单号 | 原价 | 成交价 | 买家id | 卖家id | 地区 |
|---|---|---|---|---|---|
| 1 | 密文 | 密文 | A | B | 北京 |
| 2 | 密文 | 密文 | C | B | 上海 |
| 3 | 密文 | 密文 | D | E | 杭州 |

  - Select sum(‘原价’), sum(‘成交价’) from table group by ‘地区’

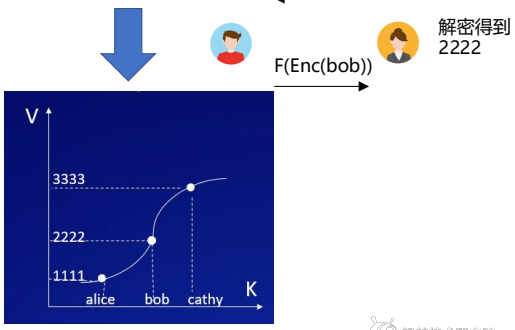
# FHE用途举例：隐私保护查询




| 姓名 | 工资 |
|---|---|
| Alice | 1111 |
| Bob | 2222 |
| ... | ... |

- 服务器对自己的数据库进行插值计算，得到多项式：y=F(x)
- 客户发送同态加密的x
- 服务器在x上同态的计算F
- 客户解密F得到y

| Key | Value |
|---|---|
| Alice | 1111 |
| Bob | 2222 |
| Cathy | 3333 |





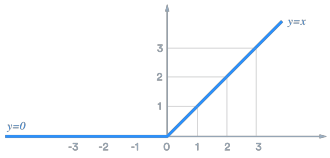

- Devil in details..
  - 数据较多时，多项式曲线次数过高怎么办
  - 数据如何编码才能高效进行多项式计算
  - 这么大的多项式用什么数据结构存
  - 如何平衡计算量和通信量
  - 目标：最终实现百万级数据秒级查询

**[ACM CCS 2018]** Labeled PSI from Fully Homomorphic Encryption with Malicious Security

# 问题: 目前只能加密计算加法和乘法



- ReLU怎么办？Softmax怎么办？




$$softmax(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$

- 可以用高阶多项式近似，但是代价巨大

# TFHE （toy example）



- 加密：
  - Enc(m) = ($\vec{a}$, b)
  - b=Δm-$\vec{a}\cdot\vec{s}$ +e
- TFHE Programmable bootstrapping （PBS）：
  - Input: Enc(m)
  - Output: Enc( f(p) )
    // p=Δm+e
- 首先构造查找表（LUT）
  - $A_0$ = f(0) + f(1) x + f(2) $x^2$ + ⋯ + f(p) $x^p$ + ⋯ f(n-1) $x^{n-1}$
- 基本思想：将A乘Enc($x^{-p}$)，然后提取出常数项
  - 注意到p=b+ $\vec{a}\cdot\vec{s}$
  - Client提供$S_1$= Enc($s_1$)作为PBS key
  - $A_0$=$A_0$*$x^{-b}$
  - $A_1$=$A_0$*(1-$S_1$)+$A_0$*$x^{-a_1}$*$S_1$= Enc($A_0$*$x^{-a_1 s_1}$)
  - 重复上述步骤即可达到目标

- TFHE可以通过查找表支持任意函数的加密计算
  - 但是不能SIMD编码，因此加法和乘法的吞吐量不如BFV

| | 多项式运算? | 非多项式运算? |
|---|---|---|
| A类全同态加密 (BFV/BGV/CKKS) | 支持packing，性能好 | 不支持，只能近似 |
| B类全同态加密 (TFHE/FHEW) | 不支持packing，性能差 | 支持 |


**[SP 2021]** PEGASUS: Bridging Polynomial and Non-polynomial Evaluations in Homomorphic Encryption.

蚂蚁技术研究院 Ant Research

- TFHE虽然可以执行任意计算…
  - 但是最多一次支持6~7个bit
    - 更高的精度需要更长的查找表，导致计算复杂度指数增长
  - 每秒最多算~100个
    - 要迭代N次密文乘法才能完成一个查找表计算
- ResNet里面的ReLU动辄百万个起步，精度要求32bit以上
- 需要新的技术：Secure Multiparty Computation （MPC）

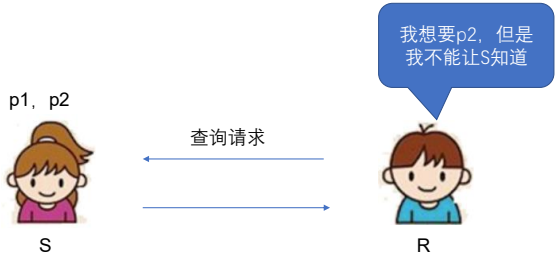
蚂蚁技术研究院 Ant Research

# Outline

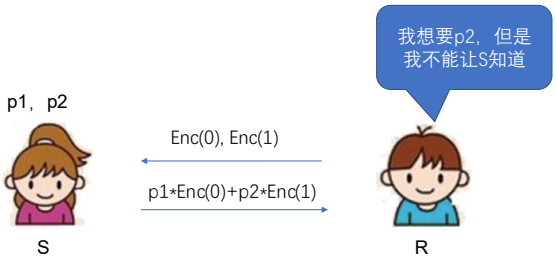
- （全）同态加密 FHE
- 安全多方计算MPC
- 我们的工作简介

蚂蚁技术研究院 Ant Research

- 重要的MPC工具1：Oblivious Transfer （OT）
  - S持有两个数p1和p2
  - R希望得到其中一个，且S不能知道R选择的是哪个



- 重要的MPC工具1：Oblivious Transfer （OT）
  - S持有两个数p1和p2
  - R希望得到其中一个，且S不能知道R选择的是哪个




蚂蚁技术研究院 Ant Research

- 重要的MPC工具2：Secret Sharing
  - 秘密共享的x记为[x]

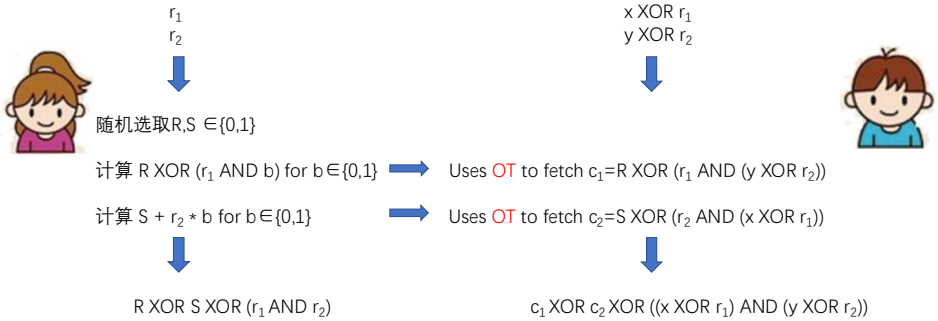

# 给定 [x],[y], 可以计算[x + y]

蚂蚁技术研究院 Ant Research


不需要网络交互

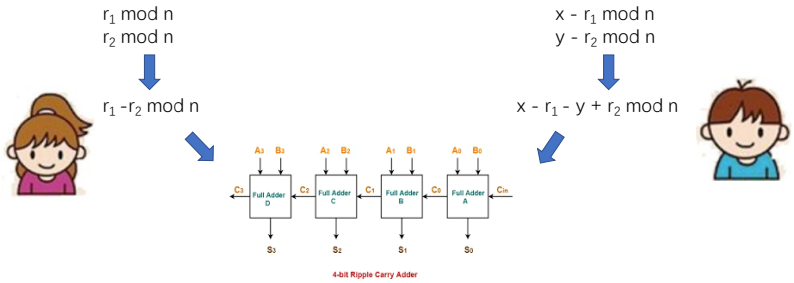
# 给定 [x],[y], 可以计算[x AND y] //n=2

蚂蚁技术研究院 Ant Research



# 给定 [x],[y], 可以计算[Relu(x)]

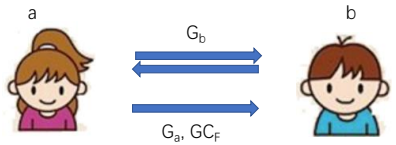
- 例: 先计算 [sign(x-y)]　// Relu(x)=x*(1-sign(x))




给定 [x],[y], 可以计算任意[f(x)]。但是每个AND门都要用OT通信一次 ⊗

# 混淆电路：2轮通信计算任意函数

- Overview：甲有a，乙有b，希望安全的计算F(a,b)
  - 甲设计混淆规则G
  - 乙使用OT，从甲处获取b的混淆值Gb
  - 甲把a的混淆值Gb，函数F的混淆电路$GC_F$发给乙
  - 乙计算$GC_F$ ($G_a$,$G_b$) = F(a,b)




$GC_F$ ($G_a$,$G_b$) = F(a,b)

- 考虑一个AND门

| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

→

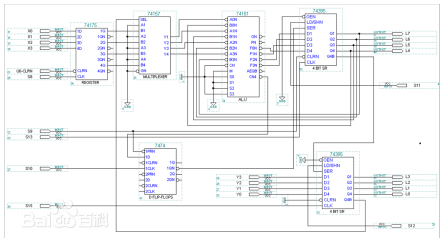
| | | |
|---|---|---|
| a | A | C |
| a | B | C |
| b | A | C |
| b | B | D |

- 混淆后的$GC_{AND}$

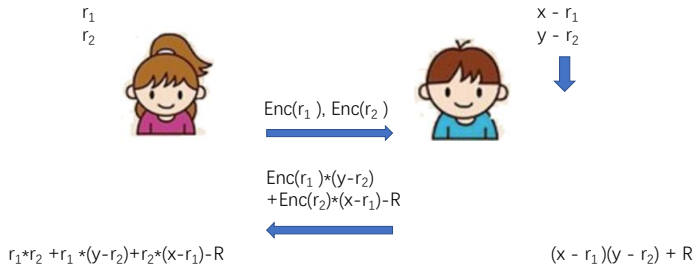
| |
|---|
| $Enc_{aA}(C)$ |
| $Enc_{aB}(C)$ |
| $Enc_{bA}(C)$ |
| $Enc_{bB}(D)$ |

# 任意函数都可通过XOR和AND门表示

- 给定 [x],[y], 可以计算任意[f(x)]
- 但是整数乘法用二进制门表示不太高效
  - 一个乘法包含大量的AND门。。



# 给定 [x],[y], 使用同态计算[x * y]



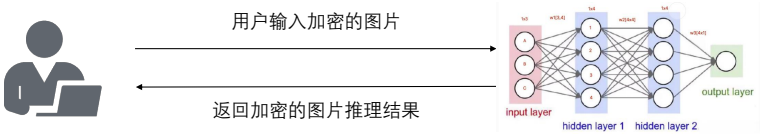

# 总结: MPC与FHE

| | 通信量 | 通信轮数 | 计算量 | 功能 |
|---|---|---|---|---|
| OT-MPC | 中 | 高（一个AND门通信一次） | 低 | 任意函数 |
| GC-MPC | 高（一个AND门几百bit） | 低 | 低 | 任意函数 |
| SIMD FHE | 低 | 低 | 中 | 受限：（加法、乘法only） |
| TFHE | 低 | 低 | 高（一个门消耗几百次乘法） | 任意函数 |

# Outline

- （全）同态加密 FHE
- 安全多方计算MPC
- 我们的工作简介

# 神经网络

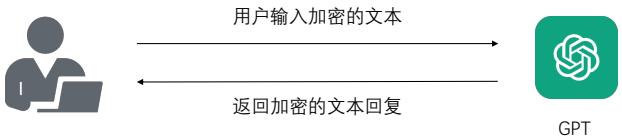



云端神经网络模型

- 线性层使用BFV同态
- 非线性层(ReLU)使用OT based MPC
- 一次Imagenet + ResNet50推理耗时80秒

**[USENIX Security 2022]** Cheetah: Lean and Fast Secure Two-Party Deep Neural Network Inference.

# 大模型




- 线性层使用BFV同态
- 非线性层(GeLU,Softmax)使用Secret Sharing based MPC
- Llama-7B输出一个token耗时13分钟

**[NDSS 2025]** Bumblebee: Secure two-party inference framework for large transformers
**[USENIX Security 2025]** GraphAce: Secure Two-Party Graph Analysis Achieving Communication Efficiency

# 跨公司的联合建模

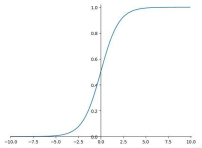
| ID | 支出 | 收入 | 车险出险次数 | 医保报销次数 |
|---|---|---|---|---|
| Alice | $50,000 | $150,000 | 1 | 3 |
| Bob | $35,000 | $90,000 | 2 | 1 |
| Cathy | $45,000 | $70,000 | 3 | 3 |
| David | $72,000 | $300,000 | 2 | 4 |

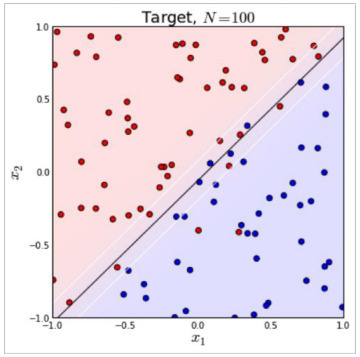
银行知道这部分　　保险公司知道这部分

- 现在Alice需要向银行申请一笔贷款
  - 银行应该批多大额度？多少利息?
  - 银行如果能拿到保险公司的数据，评估会更加精确
  - 但是直接卖数据会伤害用户隐私

# 线性模型建模（逻辑回归、推荐系统）

- 密态梯度下降优化
- 密态求矩阵乘、Sigmoid
- 亿级数据，小时级完成建模

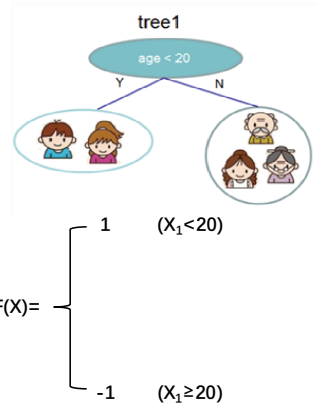





**[KDD 2021]** When homomorphic encryption marries secret sharing:
Secure large-scale sparse logistic regression and applications in risk control
**[TDSC 2021]** More efficient secure matrix multiplication for unbalanced recommender systems

# 基于决策树的密态建模 (GBDT/XGBoost)

| ID | $X_1$ (Age) | $X_2$ (Use computer daily) | Y (Like computer game) |
|---|---|---|---|
|  | 6 | Y | 1 |
|  | 12 | N | -1 |
|  | 30 | Y | 1 |
|  | 65 | N | -1 |
|  | 70 | N | -1 |

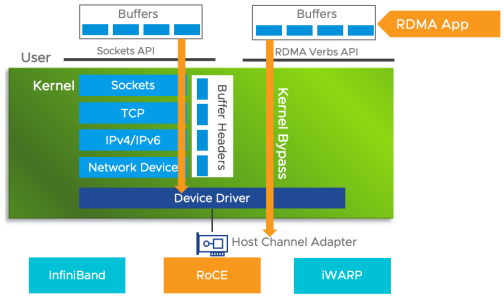



- 密态下算熵求分裂点
- 百万数据分钟级建模

**[USENIX Security 2023]**
Squirrel: A Scalable Secure Two-Party Computation Framework for Training Gradient Boosting Decision Tree.

# 硬件加速MPC

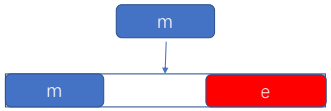
- MPC需要大量的网络传输
- RDMA网卡可以跳过TCP/IP，直接拷贝内存
  - 提高网络带宽，降低传输延迟



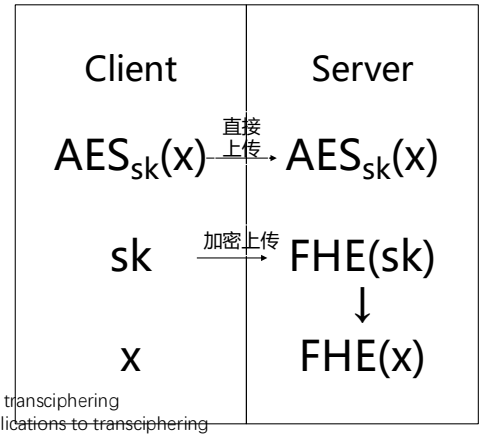

**[USENIX Security 2024]** Accelerating Secure Collaborative Machine Learning with Protocol-Aware RDMA

# 转密技术：降低FHE密文体积

- FHE会带来密文膨胀 ☹



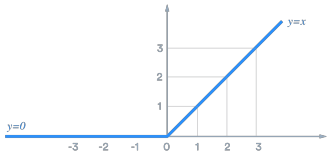

- 解决方案：
  - 先用AES等对称算法加密
  - 然后（不解密的）把AES密文转换成FHE密文




**[CHES 2025]** SoK: FHE-friendly symmetric ciphers and transciphering
**[CHES 2025]** XBOOT: Free-xor gates for ckks with applications to transciphering

# 更高的MPC安全级别

- 此前的描述都是假设大家会遵从协议
- What if.. 恶意用户不遵从协议，故意传输错误的数据？
  - 最好的结果：协议会出错终止
  - 最坏的结果：不知情的另一方的隐私可能会泄露！
- 例：一方故意给自己的secret share加上一个很大的值
  - 令ReLU(x)永远=x




# Malicious security (two party)

- 流行的解决方案需要在双方的secret share上加上验证码 (MAC)
  - 这样任意一方使用错误的数据就会被发现
- 所有的数据都加MAC，这带来了巨大的通信开销

| message | MAC |
|---|---|
| message | MAC |
| message | MAC |
| message | MAC |
| message | MAC |

| message' | |
|---|---|
| message' | |
| message' | MAC |
| message' | |
| message' | |

- 使用RMFE技术，一批数据经过转换后只需要使用一份MAC

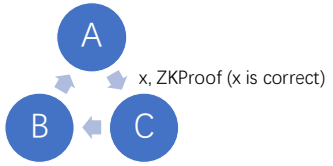
**[Asiacrypt 2023]** Degree-D Reverse Multiplication-Friendly Embeddings: Constructions and Applications
**[ACM CCS 2024]** Coral: Maliciously Secure Computation Framework for Packed and Mixed Circuits

# Malicious security (two party)

- FHE中，计算方返回加密计算结果时，需要自证计算结果是对的
  - 直接使用零知识证明ZKP可以解决，但是一般ZKP(FHE)比FHE要贵1000倍以上…
  - FHE比明文已经要贵N倍…
- 提出一种新的ZKP(FHE)方案，比FHE贵100倍以内

**[SP 2025]** ZHE: Efficient Zero-Knowledge Proofs for HE Evaluations

# Malicious security (three party)




- 三方计算中因为有两个交叉验证者，因此ZKP代价相对低

**[USENIX Security 2023]** Efficient 3PC for Binary Circuits with Application to Maliciously-Secure DNN Inference
**[ACM CCS 2024]** Sublinear Distributed Product Checks on Replicated Secret-Shared Data over  without Ring Extensions

# Higher security with less cost (PVC)

- 为了检查恶意攻击需要付出大量的自证、检验代价
- 可以进行一定的安全妥协，仅以X%的概率检查出恶意攻击
  - 虽然达不到100%，但是“可能会被查到”这个威慑很多时候已经足够

- 例：50%威慑力的MPC仅比semihonest MPC慢60%
  - 而Malicious（100%检测）MPC比semihonest MPC慢1个量级以上

**[Eurocrypt 2019]** Covert security with public verifiability: Faster, leaner, and simpler

# Open-source projects

- Production-level projects
  - 隐语: https://github.com/secretflow

- Research projects
  - https://github.com/antcplab

蚂蚁技术研究院
Ant Research

Thanks

AntCPLab.github.io