数据来源： IEEE Security and Privacy 2019 SOK

ABY是一个安全多方计算框架，是由姚电路、布尔电路、算数电路协议构成

and execute secure computation protocols. We consider eleven systems: EMP-toolkit, Obliv-C, ObliVM, TinyGarble, SCALE-MAMBA (formerly SPDZ), Wysteria, Sharemind, PICCO, ABY, Frigate and CBMC-GC. We evaluate these systems on a range of criteria, including language expressibility, capabilities of the cryptographic back-end, and accessibility to developers. We advocate for improved documentation of MPC frameworks, standardization within the community, and make recommendations for future directions in compiler development. Installing and running

Recommendation: ABY provides a powerful, low-level cryptographic interface that gives the developer significant control over performance. ABY is targeted at users who are familiar with MPC protocols and the circuit model of computation. We recommend it to developers with sufficient cryptographic background.

适合开发人员对底层进行优化开发

导入

1981年， Michael O. Rabin提出不经意传输协议：

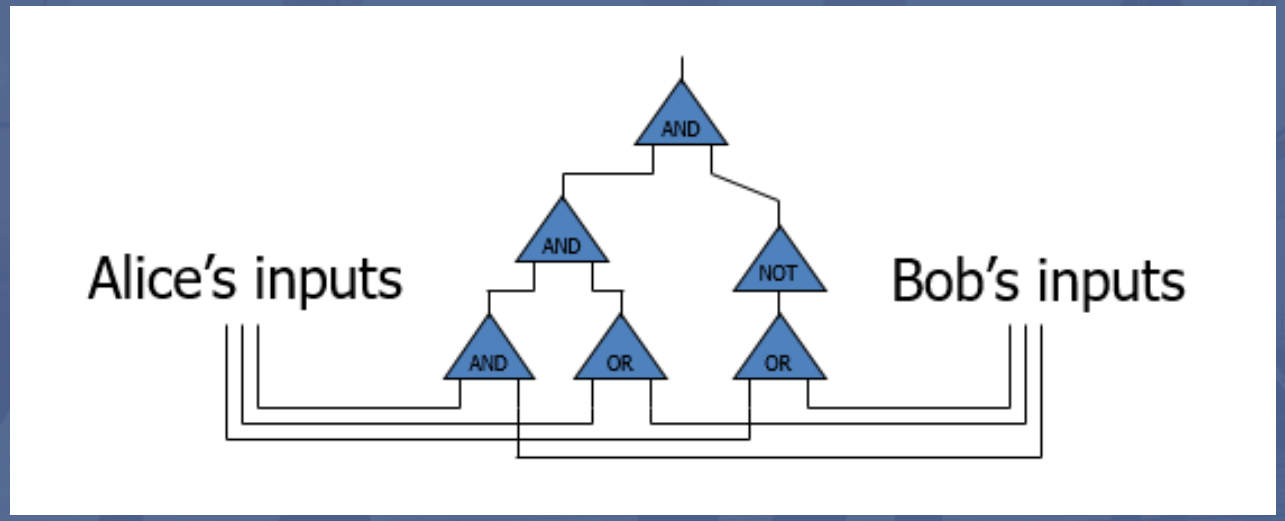
事件：A有一个秘密m，想以1/2概率传送给B

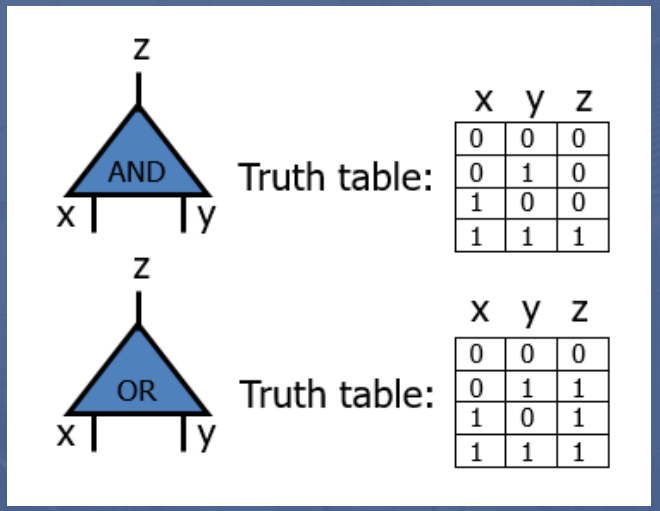
- 协议执行结束
- B知道自己是否获得了消息m
- A不知道B是否获得消息m

可用于：

- 安全计算 零知识证明 承诺 等

电路





明文计算--将函数转化为二进制布尔电路

如果安全计算每一个门→进而扩展到整个电路
那就可以安全的计算所有函数！！！

■ 导入

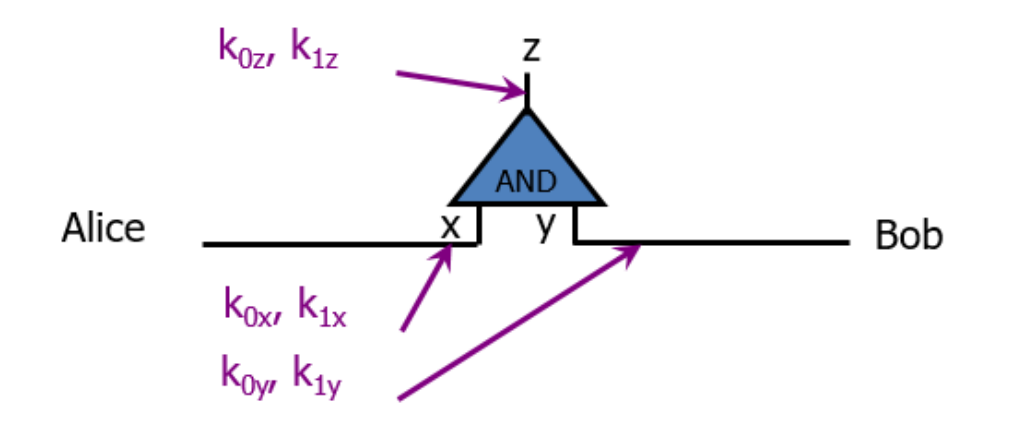
两人分粥公平方案：

Alice：负责分粥

Bob：第一个进行挑选

安全多方计算可不可以进行类似操作?

■ 姚（混淆）电路：可安全的计算任意函数
诚实好奇模型



Alice 为每一个线路选择两个随机密钥

- 一个表示“0”，另一个表示“1”
- 一个门共两个输入线路，一个输出线路，共6个密钥

Original truth table:

alice	bob	z
0	0	0
0	1	0
1	0	0
1	1	1

Encrypted truth table:

$E_{k0x}(E_{k0y}(k_{0z}))$

$E_{k0x}(E_{k1y}(k_{0z}))$

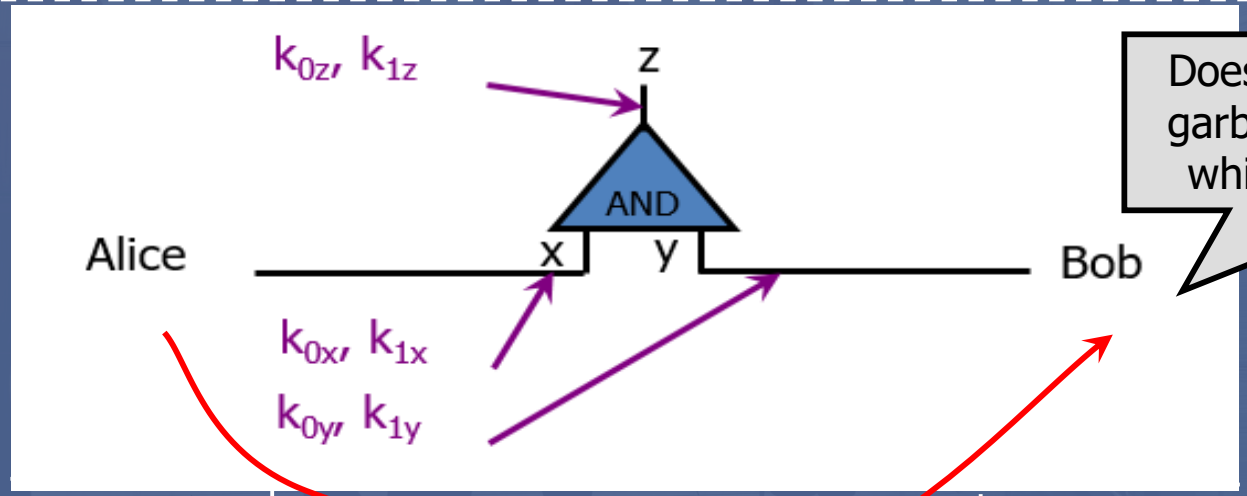
$E_{k1x}(E_{k0y}(k_{0z}))$

$E_{k1x}(E_{k1y}(k_{1z}))$

Alice 加密真值表，按照洋葱式层层加密。
最内层为输出秘钥
而后右线路输入
最后左线路输入

■ 姚（混淆）电路：可安全的计算任意函数
诚实好奇模型

Alice **随机置换**（打乱顺序）加密真值表，而后发送给Bob



Encrypted truth table:

$E_{k0x}(E_{k0y}(k_{0z}))$

$E_{k0x}(E_{k1y}(k_{0z}))$

$E_{k1x}(E_{k0y}(k_{0z}))$

$E_{k1x}(E_{k1y}(k_{1z}))$

发送

Garbled truth table:

$E_{k1x}(E_{k1y}(k_{1z}))$

$E_{k1x}(E_{k0y}(k_{0z}))$

$E_{k0x}(E_{k0y}(k_{0z}))$

$E_{k0x}(E_{k1y}(k_{0z}))$

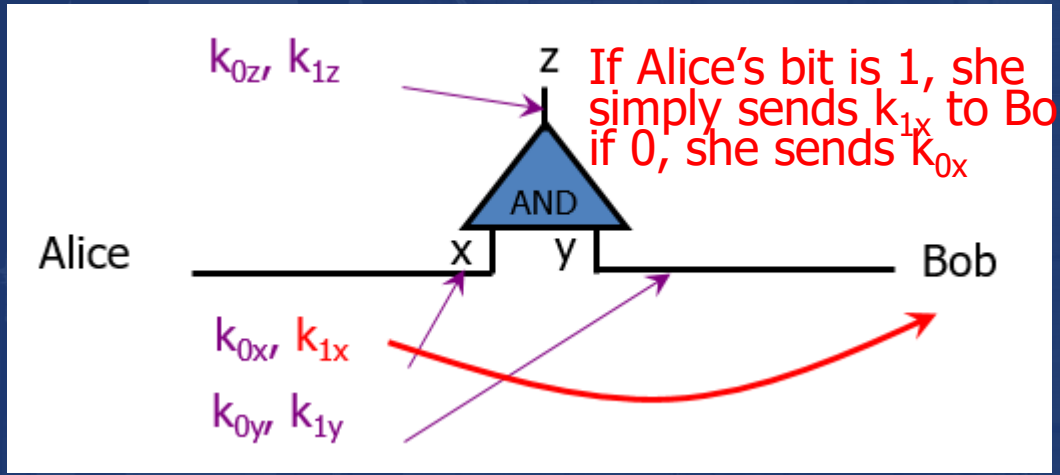
■ 姚（混淆）电路

- 可安全的计算任意函数
- 诚实好奇模型

Alice 发送自己的输入对应的密钥，因密钥是随机数，因而Bob并不知道Alice的真实输入bit

Garbled truth table:

$E_{k1x}(E_{k1y}(k_{1z}))$
$E_{k1x}(E_{k0y}(k_{0z}))$
$E_{k0x}(E_{k0y}(k_{0z}))$
$E_{k0x}(E_{k1y}(k_{0z}))$



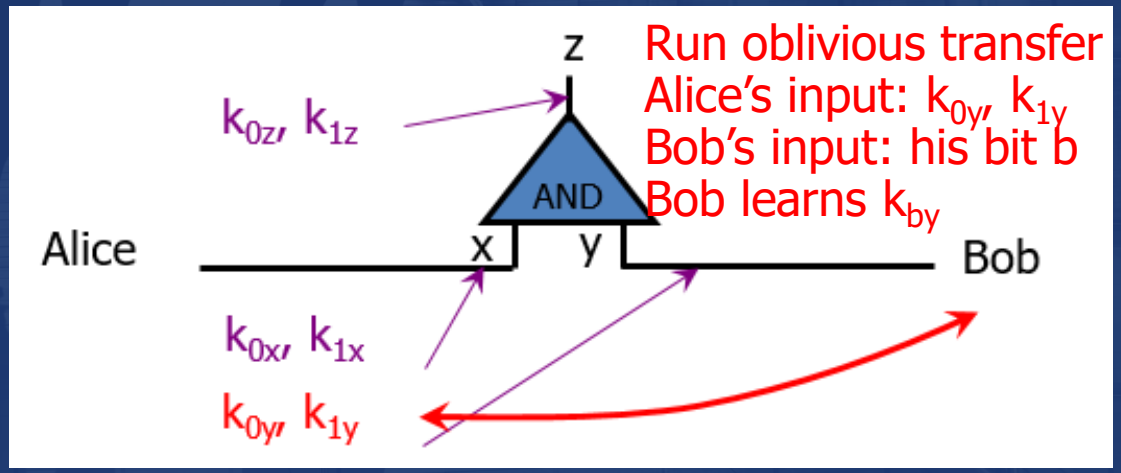
姚（混淆）电路

- 可安全的计算任意函数
- 诚实好奇模型

2选1不经意传输协议：服务器端有两个信息，客户端与服务端执行不经意传输协议之后，客户端索回自己需要的信息，服务端不清楚具体传输了哪条信息

Garbled truth table:

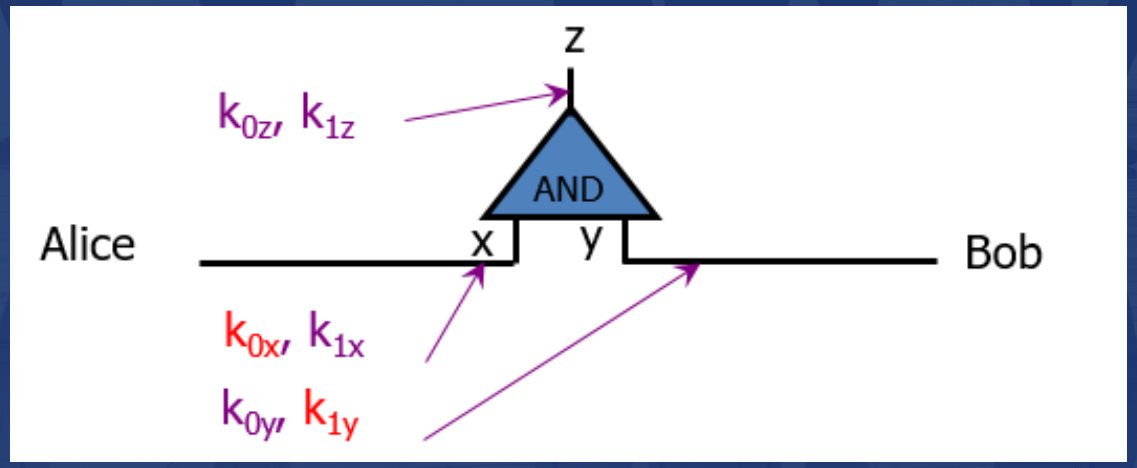
$E_{k1x}(E_{k1y}(k_{1z}))$
$E_{k1x}(E_{k0y}(k_{0z}))$
$E_{k0x}(E_{k0y}(k_{0z}))$
$E_{k0x}(E_{k1y}(k_{0z}))$



■ 姚（混淆）电路：可安全的计算任意函数
：诚实好奇模型

■给定左输入右输入密钥, Bob 解密获得输出密钥

■输出密钥是随机数，Bob并不知道输出对应的是0还是1



Garbled truth table:

$E_{k_{1x}}(E_{k_{1y}}(k_{1z}))$

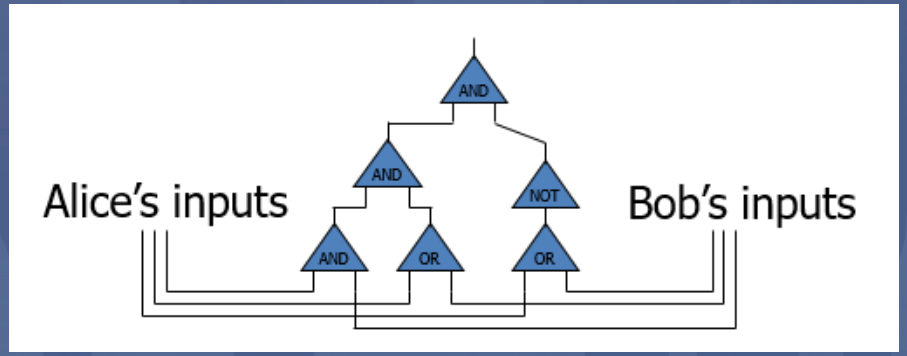
$E_{k_{1x}}(E_{k_{0y}}(k_{0z}))$

$E_{k_{0x}}(E_{k_{0y}}(k_{0z}))$

$E_{k_{0x}}(E_{k_{1y}}(k_{0z}))$

■ 姚（混淆）电路总结

- 以此类推，Bob计算整个电路



- Bob告诉Alice最终输出结果
- Alice查表获得最终输出，反馈给Bob

总结：

- 函数需转换为布尔电路形式实际中，很多函数电路非常巨大，上百万甚至上亿。
- m个门，n 个输入，需 4m 次加密和n次不经意传输。
- 在诚实好奇模型下，姚电路能够固定交互轮数完成任意函数计算，交互次数取决于输入个数。

■ 导入

- 能不能双方都参与计算？（对等地位）

布尔电路

- 可安全的计算任意函数
- 诚实好奇模型

秘密共享：
（n,t）秘密共享，将一个秘密通过门限共享方式分发给n个用户，当且仅当大于t个用户合作的时候，可以恢复秘密。

那双方怎么做?
一方掌握密钥，一方掌握密文，秘密为明文。

那两方比特共享呢?
一方掌握密钥，一方掌握密文，其中加密方式为异或操作。

异或操作具有友好特性：
满足交换律、结合律、密文可计算性。

$$\begin{aligned} c_0 \oplus c_1 &= (m_0 \oplus k_0) \oplus (m_1 \oplus k_1) \\ &= (m_0 \oplus m_0) \oplus (k_0 \oplus k_1) \\ &= M \oplus K \end{aligned}$$

对密文执行异或操作，等价于明文做异或，密钥做异或的加密。

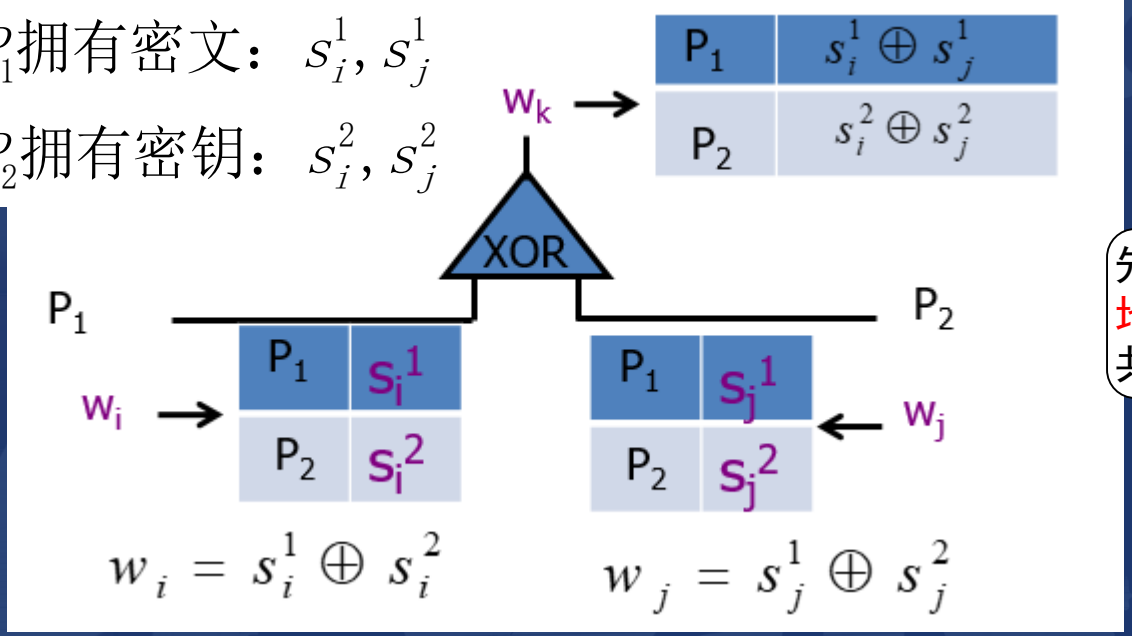
数字电路底层
与门、或门、非门集合
与门、异或门、非门集合

■ 布尔电路（异或门分析）

- 可安全的计算任意函数
- 诚实好奇模型

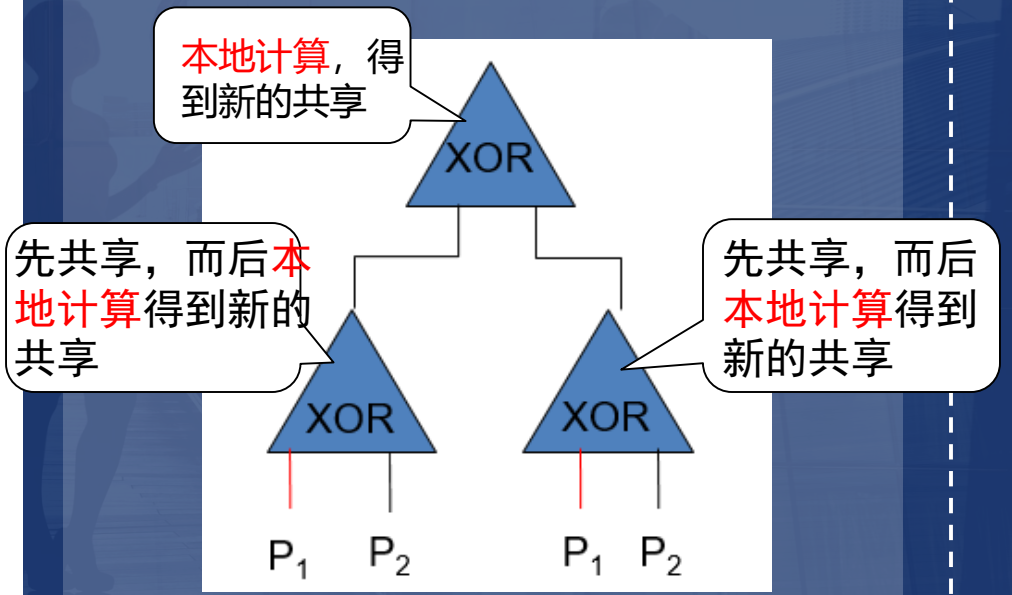
P_1拥有密文：s_i^1, s_j^1

P_2拥有密钥：s_i^2, s_j^2



$$w_i = s_i^1 \oplus s_i^2 \qquad w_j = s_j^1 \oplus s_j^2$$

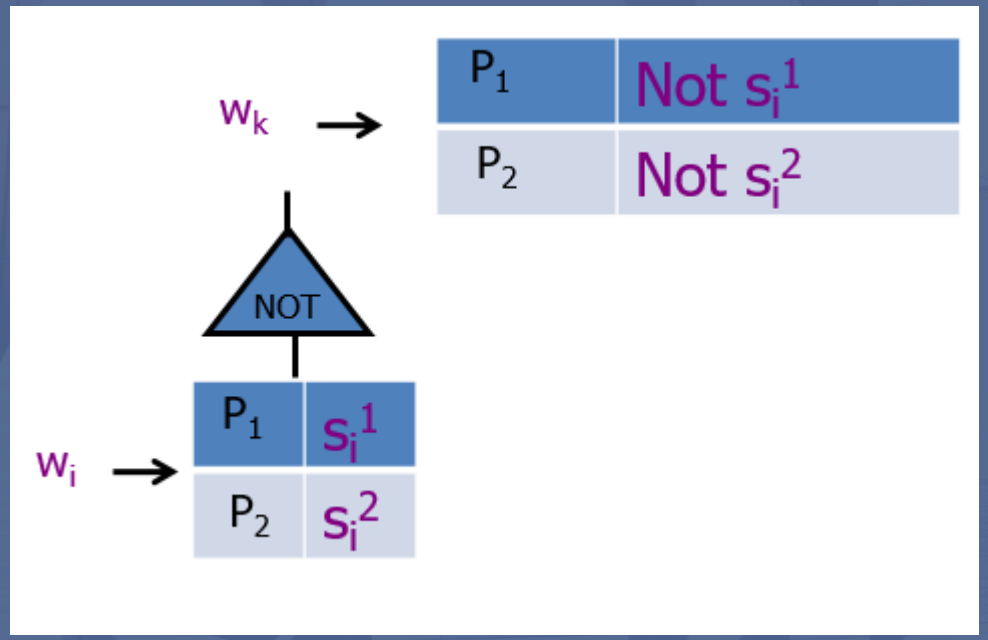
$$w_k = w_i \oplus w_j = (s_i^1 \oplus s_i^2) \oplus (s_j^1 \oplus s_j^2)$$
$$= \underbrace{(s_i^1 \oplus s_j^1)}_{P_1} \oplus \underbrace{(s_i^2 \oplus s_j^2)}_{P_2}$$



异或计算只需要在底层共享之后边可本地计算。

布尔电路（非门分析）

- 可安全的计算任意函数
- 诚实好奇模型



$$\overline{w_i} = \overline{s_i^1 \oplus s_i^2}$$

$$= \underbrace{\overline{s_i^1}}_{P_1} \oplus \underbrace{s_i^2}_{P_2}$$

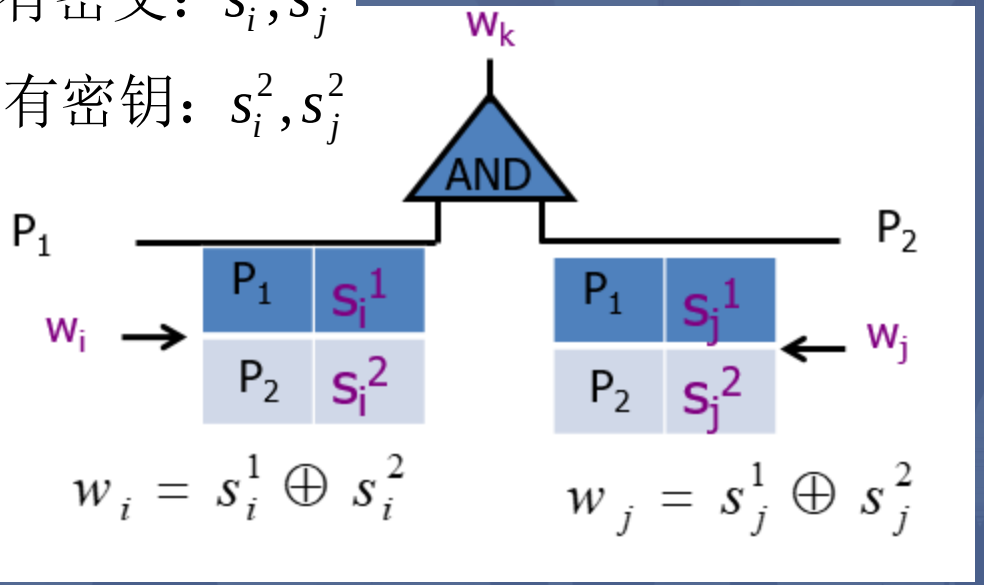
针对非门来说，思路也没问题，高效。

■ 布尔电路（与门分析）

- 可安全的计算任意函数
- 诚实好奇模型

P_1拥有密文：s_i^1, s_j^1

P_2拥有密钥：s_i^2, s_j^2



存在问题！！

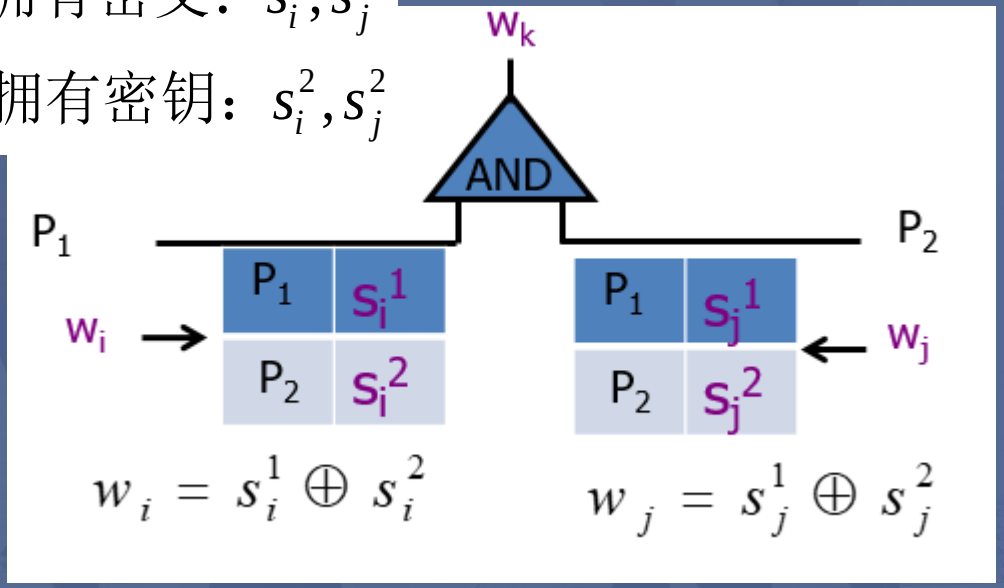
$$w_k = w_i \otimes w_j = (s_i^1 \oplus s_i^2) \otimes (s_j^1 \oplus s_j^2)$$

■ 布尔电路（与门分析）

- 可安全的计算任意函数
- 诚实好奇模型

P_1拥有密文：s_i^1, s_j^1

P_2拥有密钥：s_i^2, s_j^2



$$w_k = w_i \otimes w_j = (s_i^1 \oplus s_i^2) \otimes (s_j^1 \oplus s_j^2)$$

P_1选择一位随机比特r本地存储，并执行以下操作

$$r \oplus F_{(s_i^1, s_j^1)}(s_i^2, s_j^2) = \begin{cases} r \oplus F_{(s_i^1, s_j^1)}(0,0) \\ r \oplus F_{(s_i^1, s_j^1)}(0,1) \\ r \oplus F_{(s_i^1, s_j^1)}(1,0) \\ r \oplus F_{(s_i^1, s_j^1)}(1,1) \end{cases}$$

P_2与P_1执行1-out of-4不经意传输协议，获得

$$r \oplus F_{(s_i^1, s_j^1)}(s_i^2, s_j^2)$$

布尔电路总结

- **可安全的计算任意函数**
- **诚实好奇模型**

1. 布尔电路虽然在异或门与非门有巨大优化，但是由于与门的不经意传输变成1-out of-4，所以综合对比起来姚电路（所有门1-out of-2）来说效率差不多。
2. 布尔和姚电路都是对二进制操作友好的电路。
3. 布尔电路的意义在于：

- 提出一个新的角度去解决问题。
- 可扩展为代数（算数）级别计算，效率更高。
- 可以通过（n，t）秘密共享结合，可以拓展支持多方计算。

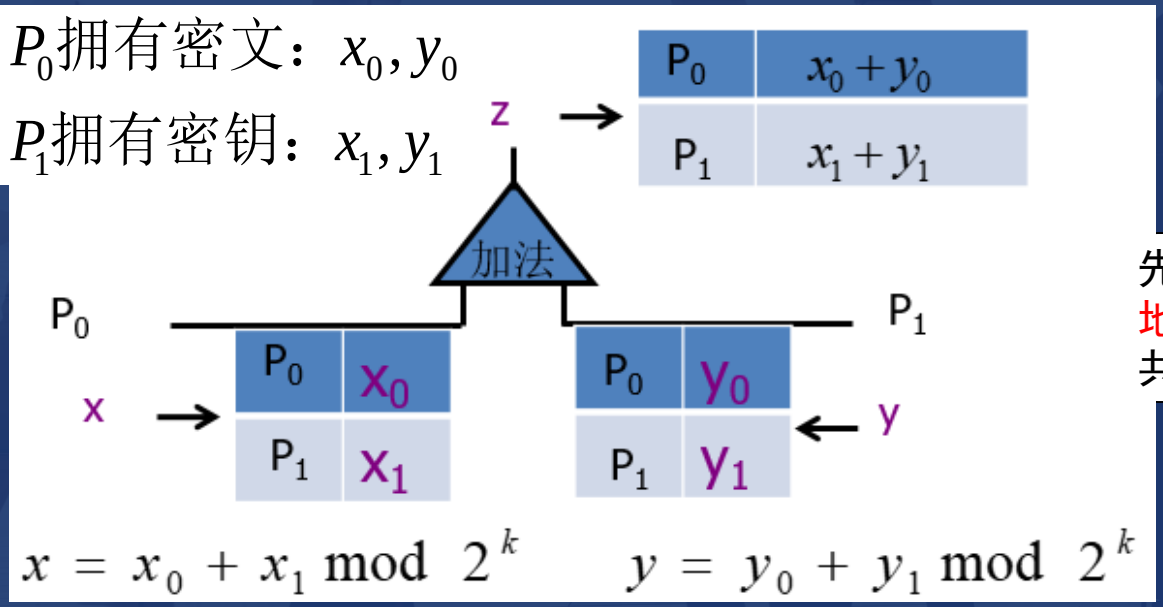
■ 导入

- 逻辑比较大小、比特之间的运算这些操作。
- 实际之中更多去用加法，乘法。能不能更优化。

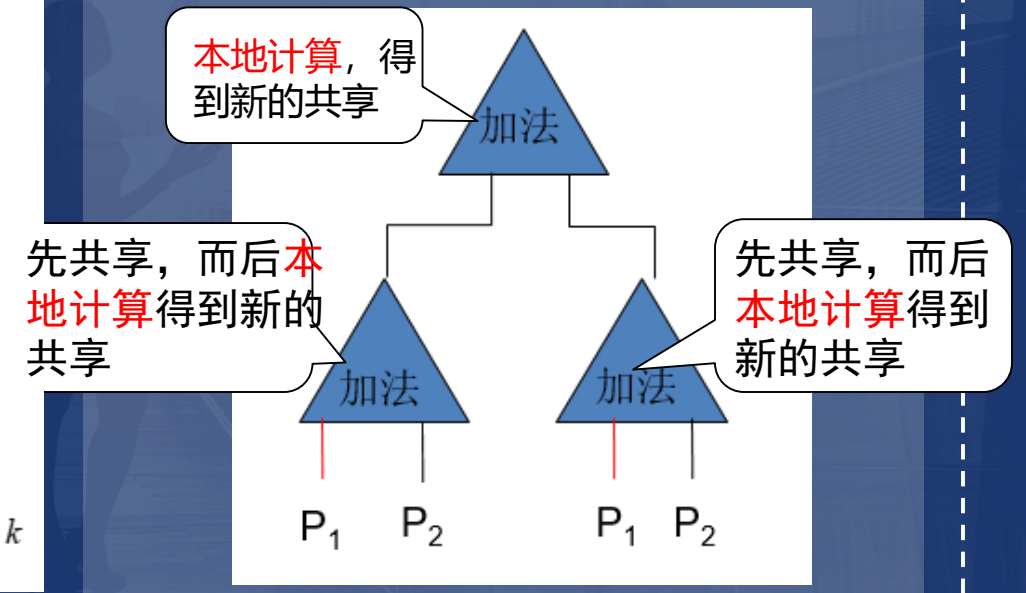
■ 算数电路（加法门分析）. 诚实好奇模型

P_0拥有密文：x_0, y_0

P_1拥有密钥：x_1, y_1



$x = x_0 + x_1 \bmod 2^k$　　$y = y_0 + y_1 \bmod 2^k$



$$z = x + y = (x_0 + x_1) + (y_0 + y_1) \bmod 2^k$$
$$= \underbrace{(x_0 + y_0 \bmod 2^k)}_{P_0} + \underbrace{(x_1 + y_1 \bmod 2^k)}_{P_1}$$

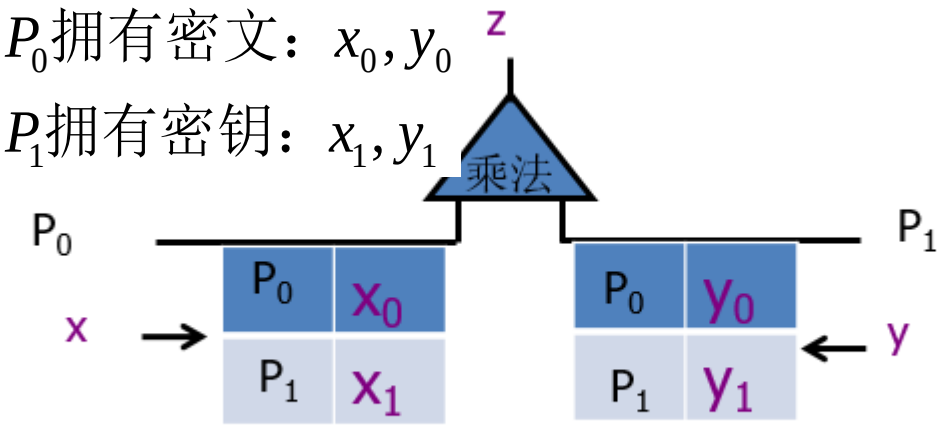
针对加法门，思路是合理的，并且非常高效。

■ 算数电路（乘法门分析）

· 诚实好奇模型

P_0拥有密文：x_0, y_0

P_1拥有密钥：x_1, y_1



$$x = x_0 + x_1 \bmod 2^k \qquad y = y_0 + y_1 \bmod 2^k$$

$$z = x \cdot y = (x_0 + x_1) \cdot (y_0 + y_1) \bmod 2^k$$

$$= \underbrace{x_0 \cdot y_0}_{P_0} + \underbrace{x_1 \cdot y_1}_{P_1} + x_1 y_0 + x_0 y_1$$

1.在布尔计算中可以通过枚举来表达所有可能。

2.算数计算中每一方都有2^k种可能，共$2^k 2^k = 2^{2k}$种可能。

3.若取128比特安全，则$2^{2k} = 2^{128}$，所以k=64。

4.所以用不经意传输方式仅仅是理论上可能。

■ 算数电路（乘法门分析）. 诚实好奇模型

解决思路：引入乘法三元组

- 乘法三元组：是独立于待计算的乘法门，可以预先计算好待后续乘法门使用(注：每个乘法三元组只能使用一次)

① P_0拥有a_0 b_0 c_0

② P_1拥有a_1 b_1 c_1

③ 满足$(a_0+a_1)(b_0+b_1)=(c_0+c_1)$

- 乘法门计算：

① P_0公开x_0+a_0，y_0+b_0

② P_1公开x_1+a_1，y_1+b_1

③ 双方均可计算获得$e=x+a, f=y+b$

④ P_0本地计算：$z_0=-fa_0-eb_0+c_0 \bmod 2^k$

⑤ P_1本地计算：$z_1=ef-fa_1-eb_1+c_1 \bmod 2^k$

使用同态Paillier协商乘法三元组：

① P_1计算发送给P_0：

$$E(b_1),\ E(a_1)$$

② P_0计算发送给P_1：

$$E(b_1)^{a_0} E(a_1)^{b_0} E(a_0 b_0 - c_0)$$
$$= E(a_0 b_0 + a_0 b_1 + a_1 b_0 - c_0)$$

③ P_1解密并计算:

$$a_0 b_0 + a_0 b_1 + a_1 b_0 - c_0 + a_1 b_1 \bmod 2^l$$

■ 算数电路总结

1. 加法本地计算，乘法利用三元组也非常高效（注：三元组是可以预先计算好的）。
2. 布尔共享和混淆电路可以支持负数、小数等任意计算，算数电路看起来只能支持非负数的加法和乘法（**也可以利用放大操作支持**）。
3. 算数电路支持加法和乘法作为计算单元，可以扩展到减法理论上，任何函数都可以通过多项式组合的方式去**无限逼近**。
4. 通过与布尔比较大小结合，可以支持非常多的非线性计算，这个要结合具体计算函数。

ABY框架总结

- 诚实好奇模型

Table 3.1: Operations

Operations	AND	XOR	OR	ADD	MUL	SUB	GT	MUX	INV
Arithmetic	✗	✗	✗	✓	✓	✓	✗	✗	✓
Boolean	✓	✓	✓	✓	✓	✓	✓	✓	✓
Yao	✓	✓	✓	✓	✓	✓	✓	✓	✓

	姚电路	布尔共享	算数共享
支持功能	★★★★★	★★★★★	★★
计算效率	★★	★★	★★★★★
支持数据	★★★★★	★★★★★	★

1.可以利用各个电路的优点分别去做其优势的方向。

2.三个电路可以通过加掩码的方式相互转换。

3.姚电路和布尔共享优点就是支持数据与功能比较全。而算术共享是计算效率高。

机器学习算法	**逻辑/线性回归**	**DNN**	**CNN**	**GNN**	**RNN**
安全多方计算	基础算子	矩阵 矩阵加法 矩阵乘法	激励函数 Sigmoid函数 RELU 函数 Tanh 函数	基础计算 基础比较加乘 精准指数 高效除法	
	技术	秘密分享 不经意传输	混淆电路 布尔电路	同态加密 算数电路	