密码学

入门与进阶

2020.02.18

内容大纲

密码学入门

- 1. 密码简史
- 2. 哈希函数
- 3. 加密算法
- 4. 数字签名

密码学进阶

- 1. 安全多方计算
- 2. 同态加密
- 3. 零知识证明
- 4. 代理重加密

1 密码简史

密码技术的演进

隐写术

特殊的物理处理手段，使得未提前获知处理方式的人无法读取其中所蕴藏的信息

The Enigma

二战时，德军用于保密通讯的加密设备，后来被以英国和波兰为首的盟军使用密码分析技术破解，被认为将二战胜利提前了两年

公钥密码学、可证明安全
安全多方计算、量子密码

公钥密码学建立了现代信息社会的安全基石，新型密码协议的设计以及量子密码等前沿密码学理论的探索让其得以面向未来

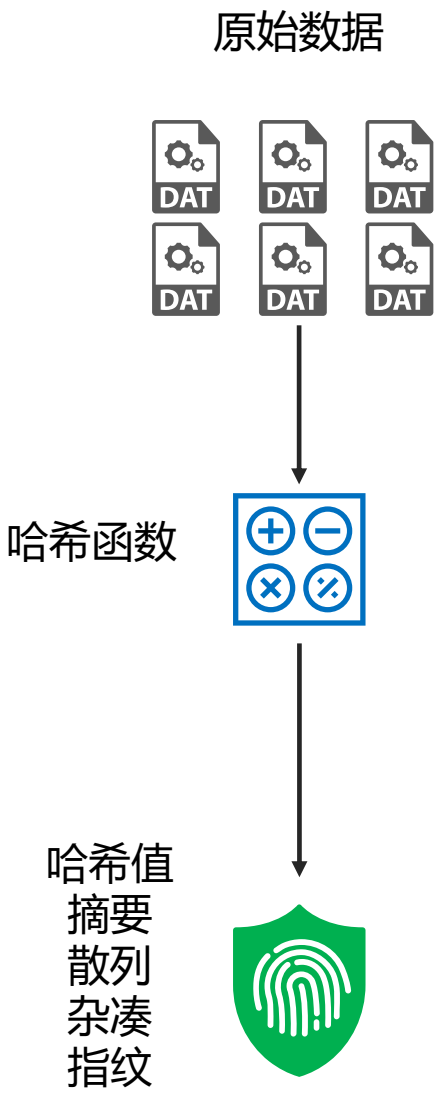
很久以前

近代

现代

2 哈希函数

哈希函数



以任意长度的数据为输入，根据不同的哈希算法输出相应固定长度的值。

性质

- 抗碰撞：无法找到两个不同的输入，使得其输出一致；
- 单向性：正向计算效率高，反向计算难度非常大；

应用

- 口令保护：现代web应用中，账户系统的口令通过哈希函数处理后，再存储于服务器上，使得口令只对用户自己可知；
- 软件保护：对外发布软件时，同时使用哈希函数计算出软件的哈希值，与软件一起发布，防止用户下载假冒软件；
- 区块链：在区块链中，账户与交易都涉及哈希函数的使用。

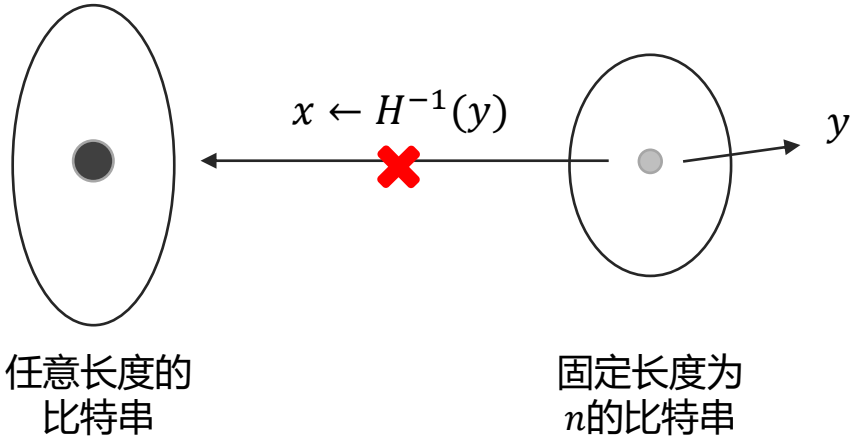
算法

- 国际：MD4, MD5, SHA1, SHA256, SHA3；
  - MD系列的哈希算法已被破解，SHA3为目前最安全的哈希函数
- 国内：SM3；
  - 国产自主研发的商用密码哈希算法

哈希函数：单向性

单向性是指对于哈希函数 $H: D \rightarrow \{0,1\}^n$，对于一个给定的哈希值，无法逆向计算出其原像输入，即：

- 给定一个哈希值 y，无法计算出 x，使得 $\mathrm{y} = H(x)$



如何找到反向计算出原像?

在输入区间中，尝试每一个可能的值 x，使得 $H(x) = y$

$$
\begin{aligned}
73\text{TH/s} &= 73 \times 10^{12}\text{/s} \\
&\approx 2 \times 10^{22}\text{/year} \\
&\approx 2^{78}\text{/year} \\
2^{160} \div 2^{78} &= 2^{82}\ \text{year} \ > \ 10^{24}\ \text{year}
\end{aligned}
$$

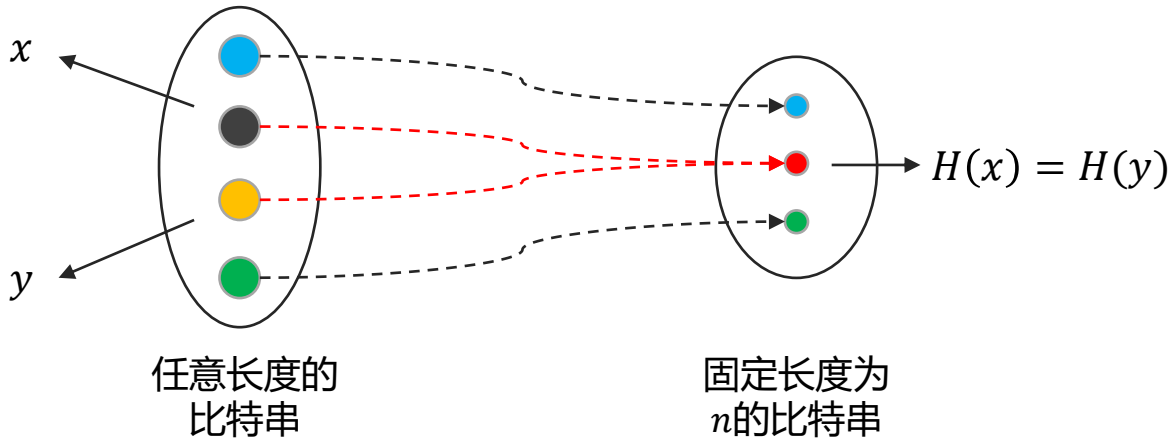
以哈希函数SHA1（输出为 160-bit）来说：
假设计算机具备的算力为73TH/s（即每秒可执行73×10^{12}次哈希，Bitmain算力最高矿机Antminer S17+），那么其一年可执行的哈希运算为 2×10^{22}次，要搜索出每一个原像大概需要 2^{82}年，超过100万亿亿年。

难度太大，几乎不可能!

哈希函数：抗碰撞

碰撞指的是对于哈希函数 $H: D \to \{0,1\}^n$，在输入区间 D中的任意两个不同输入 x、y，两者的哈希值相同，即：

- $H(x) = H(y)$
- $x \neq y$



碰撞确实存在，但如何找到?

以哈希函数SHA1（输出为 160-bit）来说：
其输出空间为 $(0, 2^{160})$，假设有1万亿个哈希值，其发生碰撞的概率仅为3×10^{-25}。

概率太低，几乎不可能!

生日攻击问题：一个班级总共有 N名同学，任意两个同学是同一天生日的概率有多大？（一年共有 $T = 365$天）

我们先计算所有人生日都不相同的概率 $p(n)$：

$$p(n) = 1 \cdot \frac{365-1}{365} \cdot \frac{365-2}{365} \cdots \frac{365-(N-1)}{365}$$
$$= (1 - \frac{1}{365}) \cdot (1 - \frac{2}{365}) \cdots \left(1 - \frac{N-1}{365}\right)$$
$$\approx e^{-\frac{1}{365}} \cdot e^{-\frac{2}{365}} \cdots e^{-\frac{N-1}{365}}$$
$$= e^{-\frac{\frac{1}{2}N(N-1)}{365}}$$
$$= e^{-\frac{N(N-1)}{2 \cdot T}}$$

泰勒展开公式
$e^x \approx 1 + x$

则至少有两个人相同生日的概率为 $q(n) = 1 - p(n)$，

$$q(n) = 1 - e^{-\frac{N(N-1)}{2 \cdot T}}$$

N为输入区间，T为取值空间

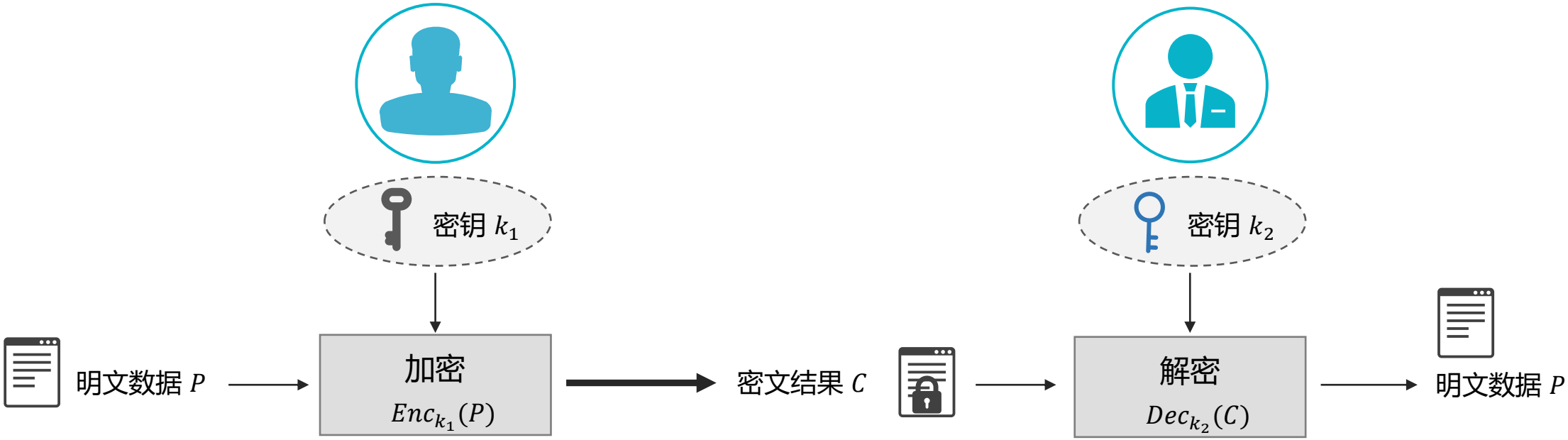
哈希碰撞的概率

3 加密算法

加密算法

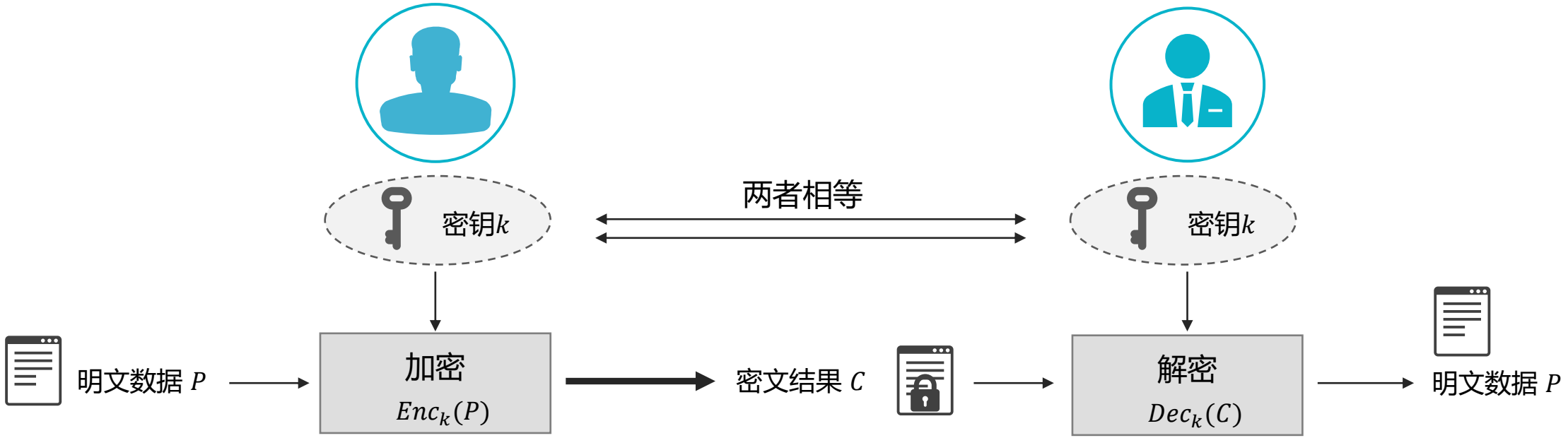
加密算法是对传递数据进行编码处理的技术手段，使得编码后的数据对于未获知具体处理方式的其他方不可读。

- 原始数据称为“明文”，编码后的数据称为“密文”；
- 将明文转换为密文的过程称为**“加密”**，从密文还原为明文的过程称为**“解密”**；
- 对明文进行加密和对密文进行解密所需要的秘密信息称为**“密钥”**。



若两个密钥相同，则称为“对称加密”；若两个密钥不同，那么称为“非对称加密”。

对称加密



对称加密是使用一把密钥对数据进行加密，获得的密文也只能由这把密钥完成解密，即加解密的密钥相同。

- 确保加密之前，密钥在双方的安全共享；
- 对称密钥通常更新频繁，需维护这些不断更新的密钥

优势：快，常用于大量数据的加密。
问题：如何找到所需的安全共享方式?

- 若双方之前未有过见面，则如何通过安全通讯传输密钥?

Diffie-Hellman密钥交换协议

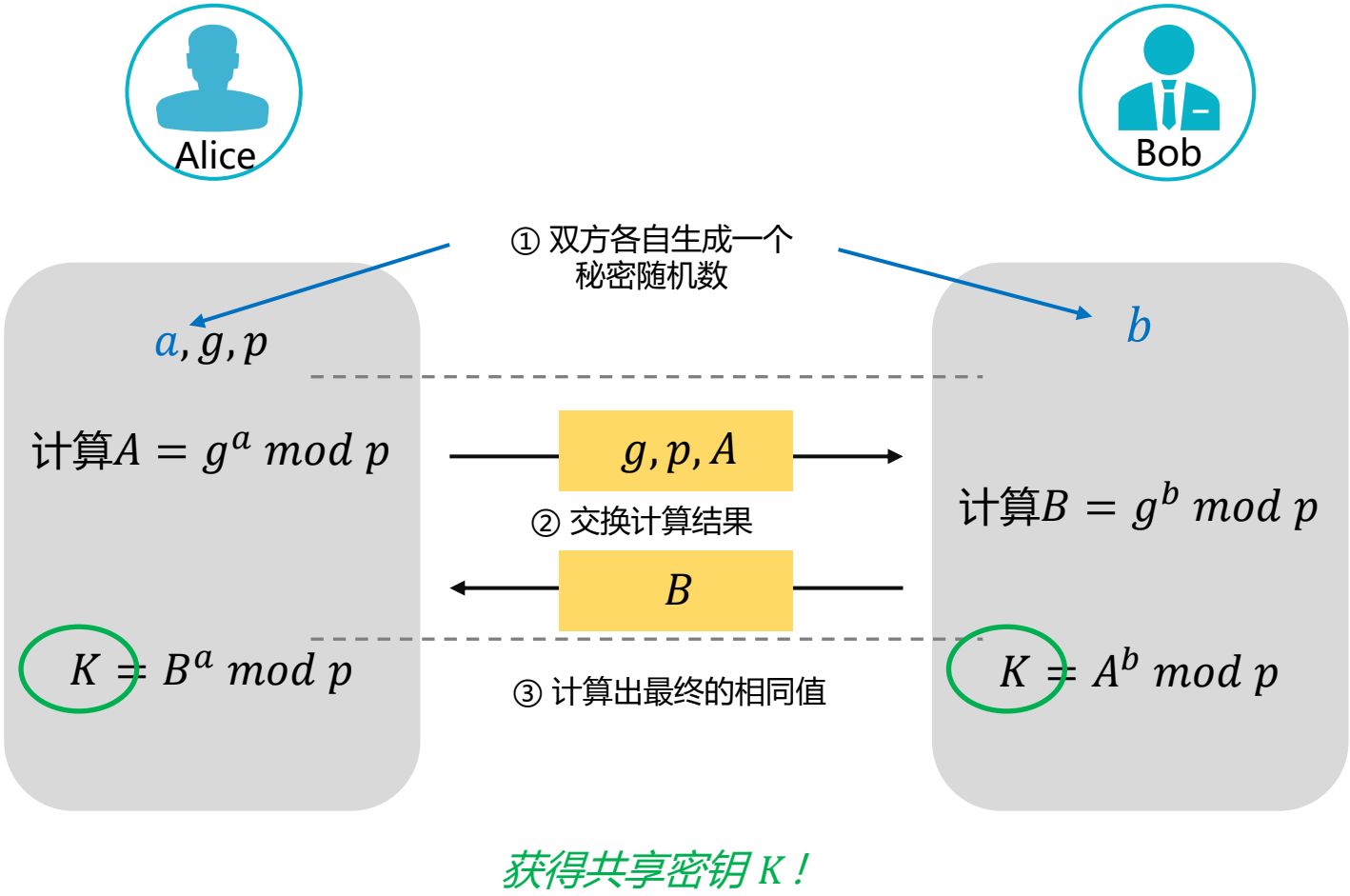
DH密钥交换协议用于在一个不安全的信道中，使得通讯双方能够生成一个共享的私钥，从而实现安全传输信息的目的。由Diffie和Hellman在1976年提出，开启了公钥密码学的篇章。

目的

通讯双方无需共享任何秘密信息，执行DH协议，即可获得一个只有双方可知的密钥。

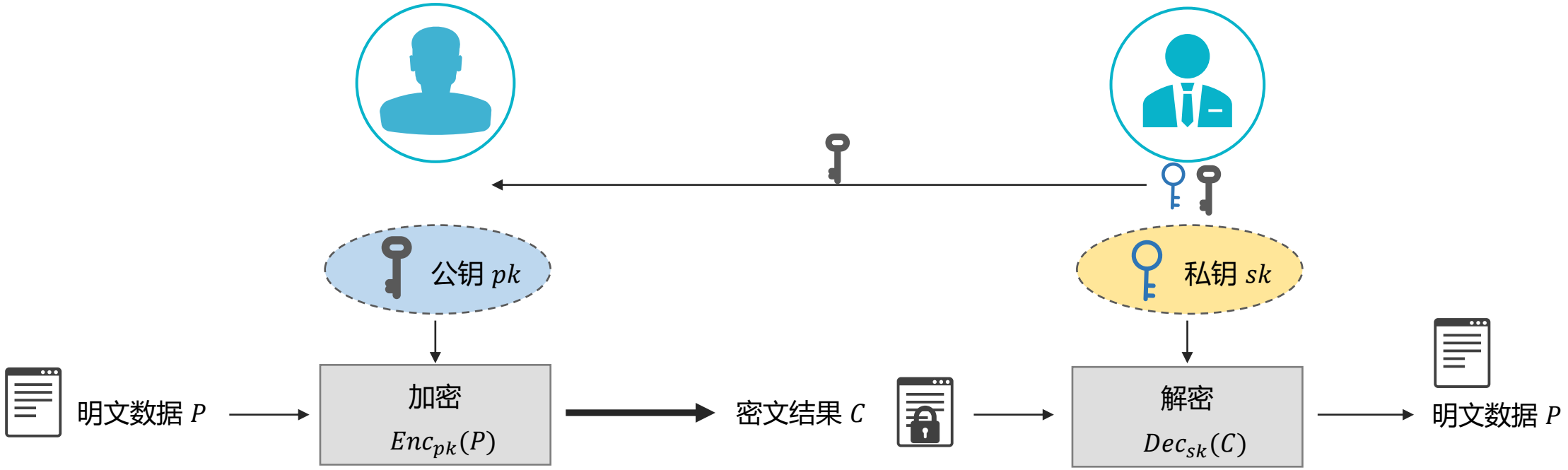
应用

DH广泛应用在当前许多协议中，比如TLS/SSL、SSH、PKI等，保障通讯安全。



获得共享密钥 K！

非对称加密



非对称加密使用两把不同的密钥分别对数据进行加密和解密，获得的密文也只能由这把与加密时所使用的密钥所对应的另一把密钥解密。

- 两个密钥通常成对生成，也称为“密钥对”、“公私钥对”；
- 加密的密钥称为“公钥”，可公开；解密所用的密钥称为“私钥”，必须保密；
- 在加密之前，公钥需提前共享

优势：更安全，无需提前将私钥共享。
劣势：慢，通常不会用于加密数据，而用来对密钥进行加密
问题：如何确保将正确的公钥发送给数据接收方？

- 通过CA对公钥进行签发，将实体身份与公钥相绑定，并由第三方认证机构进行认证

非对称加密：RSA

RSA算法

公钥：(e, N)　**私钥**：(d, N)

- 选取两个大素数p和q；
- 计算$N = pq$，以及$\varphi(N) = (p-1)(q-1)$；
- 选取另一个整数e，使得$gcd(e, \varphi(N)) = 1$；
- 计算出d，满足$ed = 1 \ mod \ \varphi(N)$。（拓展欧几里得定理）

- 对数据m的加密：$c = m^e \ mod \ N$；
- 解密：$m = c^d \ mod \ N$。

证明：$c^d \equiv m \ mod \ N$

要证$c^d = m \ mod \ N$，根据中国剩余定理，仅需证$c^d = m \ mod \ p$或$c^d = m \ mod \ q$。下面我们证明$c^d = m \ mod \ p$：

由于$c = m^e \ mod \ N$，则$c = m^e \ mod \ p$，则$c^d = (m^e)^d = m^{ed} = m^{k\varphi(N)+1} = m \cdot m^{k(p-1)(q-1)} = m \cdot (m^{p-1})^{k(q-1)} \ mod \ p$

由费马小定理，则

$$c^d = m \cdot (1)^{k(q-1)} \ mod \ p = m \ mod \ p$$

同理，$m = c^d \ mod \ q$

从而有，$m = c^d \ mod \ N$

困难问题假设

大整数分解难题：对于整数运算$N = pq$。给定(p, q)，计算出N是容易的；但给定大整数N，分解出两个素数p和q是困难的。

非对称加密：ElGamal

ElGamal加密算法

- g为群G的生成元，G的阶为q；
- 选取$k \in Z_q$；
- 计算$y = g^k$；

公钥：y　**私钥**：k

- 对数据m的加密：
  - 选取随机数r，计算$A = g^r \ mod\ q$，
    $B = m \cdot y^r \ mod\ q$，
- 对密文(A, B)的解密：
  $m = B \cdot A^{-k} \ mod\ q$

证明：$B \cdot A^{-k} \stackrel{?}{=} m \ mod\ q$

$$
\begin{aligned}
&B \cdot A^{-k} = B \cdot (g^r)^{-k} \\
&= m \cdot y^r \ (g^r)^{-k} \\
&= m \ \cdot (g^k)^r (g^r)^{-k} \\
&= m \cdot g^{kr-kr} = m \cdot g^0 \\
&= m \cdot 1 = m
\end{aligned}
$$

即$m = B \cdot A^{-k} \ mod\ q$

困难问题假设

离散对数难题：对于对数运算$A = g^r \ mod\ q$。给定(g, q, r)，计算出A是容易的；但给定(g, q, A)，计算出r是困难的。

非对称加密与对称加密相结合

对称加密：快，但传输不安全

非对称加密：慢，但安全性更高

对称加密：密钥K
非对称加密：私钥sk，公钥pk

非对称加密用于密钥传输，对称加密用来加密数据。

密钥 K

公私钥对 (sk, pk)

①发送公钥 pk

②加密对称密钥
$C_0 \leftarrow ASymEnc(pk, K)$

C_0

③解密获得对称密钥
$K \leftarrow AsymDec_{sk}(C_0)$

获得对称密钥 K

④使用对称密钥加密数据
$C_0 \leftarrow SymEnc_K(Data)$

⑤发送加密数据 C_1

⑥解密获得数据明文
$Data \leftarrow SymDec_K(C_1)$

得到数据明文 $Data$

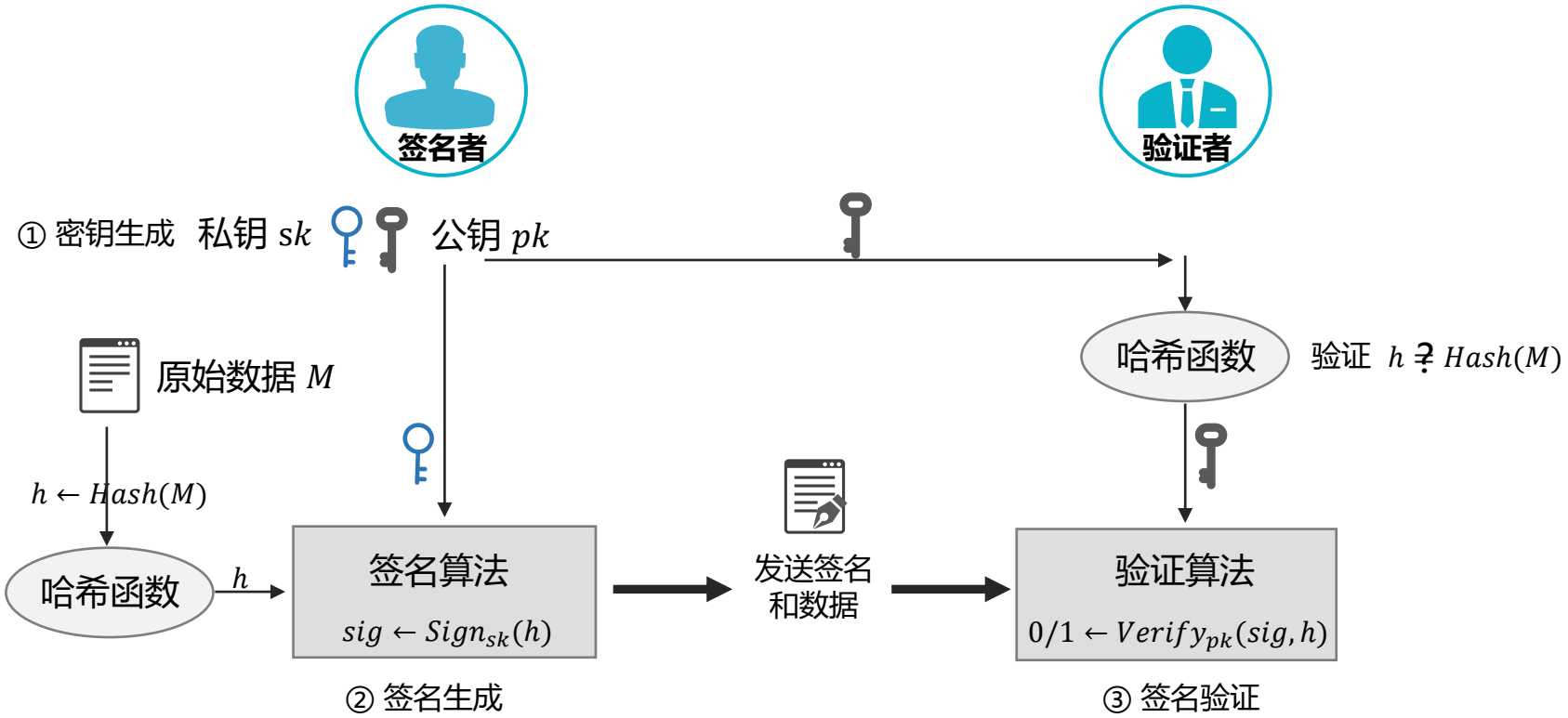
4 数字签名

数字签名

非正式定义：数字签名是验证电子信息或文件合法性的一种数学方法，用于确保电子信息的来源与完整性。

形式化定义：一个数字签名方案包含三个算法：

- $KenGen()$：密钥生成算法，生成签名者的公私钥对；
- $Sign(sk, M)$：签名算法，签名者使用其私钥对消息进行签名；
- $Verify(sig, M, pk)$：验证算法，验证者收到发送方所提供的签名后，验证签名的有效性。



应用场景：数字证书

现实世界中的证书

证书持有者证明其所声明身份的文件，由第三方权威机构签发。比如身份证、护照、驾照等，具备特点：

- 防篡改
- 由权威机构签发
- 与实际身份信息相绑定，如姓名、出生日期、证件号等

信任证书的理由

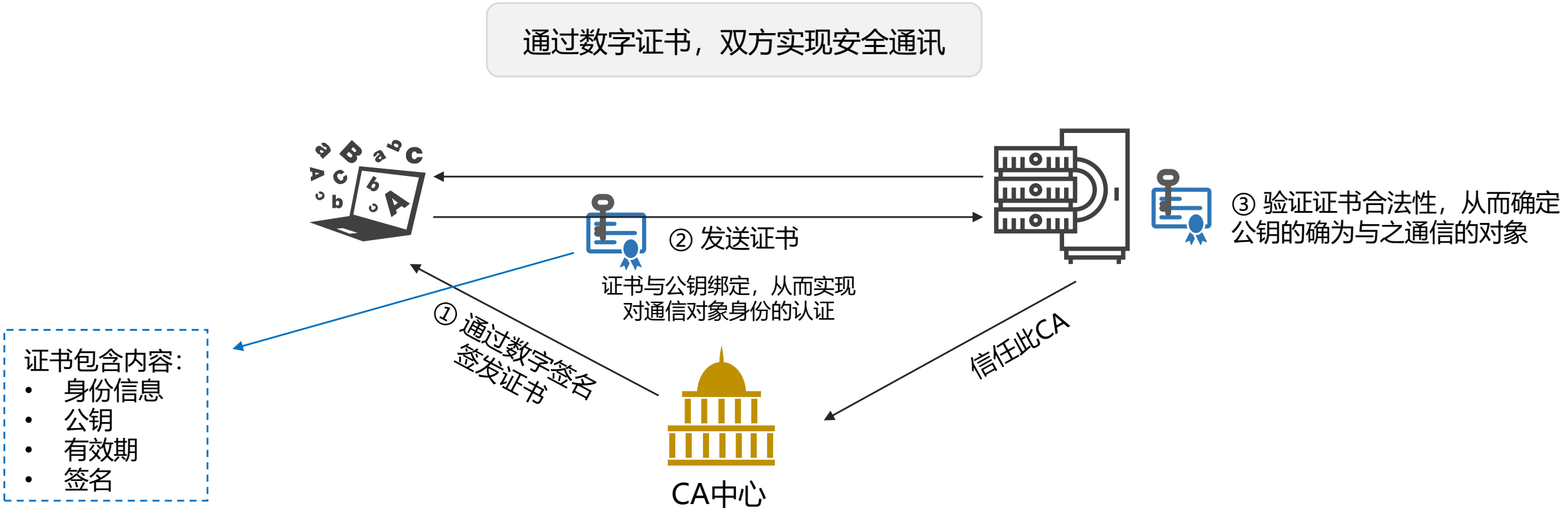
- 做伪证的难度很大
- 相信签发证书的机构

那么数字证书也需同样做到不易伪造，并且验证证书的一方也信任签发数字证书的第三方机构。

应用场景：数字证书

如何保证公钥的传输安全，即如何确保与之相通信的对象不是他人假冒？

- 物理身份与公钥相绑定
- 由CA对证书进行签名认证



密码学进阶

- 1. 安全多方计算
- 2. 同态加密
- 3. 零知识证明
- 4. 代理重加密

1 安全多方计算

场景引入

Buono

Cattivo

Brutto

场景

某地刚发现一个金矿，官方发出布告要公开拍卖此金矿的开采权，三人为了获得金矿的开采权执行拍卖，约定出价最高的一方赢得此次拍卖。但各自不愿直接公开透露具体的出价。

问题定义

三人持有各自的出价B_i，想计算出函数

$$\max(B_{Buono}, B_{Brutto}, B_{Cattivo})$$

但核心是不能互相透露各自的B_i。

即著名的“**百万富翁问题**”，由华人学者姚期智先生于1986年提出，并由此获得了计算机领域的最高奖——图灵奖。

解决方案：求于人

Buono

Cattivo

警长先生

Brutto

三人都信任警长先生，各自将出价透露给警长，警长作为一个完全可信的第三方，统计出拍卖结果。

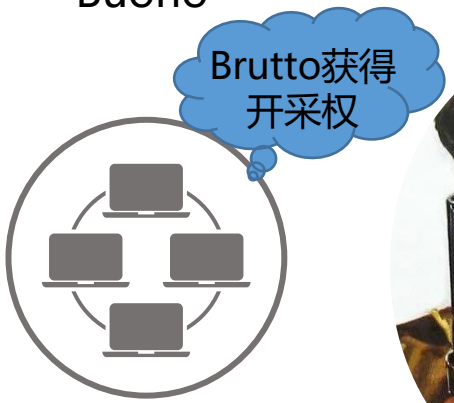
此可信第三方需满足：

- 正确执行运算过程；
- 不会向任意一方透露隐私数据

解决方案：MPC算法

Buono

Cattivo



MPC
protocol

Brutto

现实中，很难找到一个完全可信第三方，或者可能并不存在。通过MPC协议，无需信赖任何一方，以各自隐私数据为输入，计算执行完成，各方除了获得计算结果外，无法获得关于其他方隐私数据的任何有效信息。

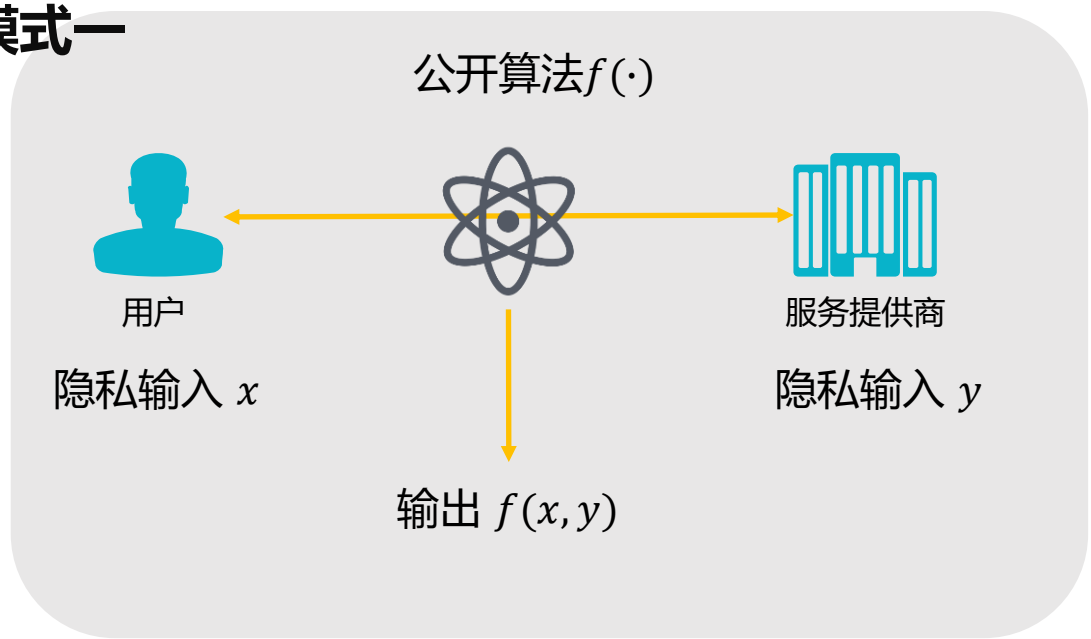
MPC算法可保证：

- **计算正确性**：计算结果与传统计算结果一致；
- **隐私性**：计算过程中，不会泄露关于隐私数据的信息

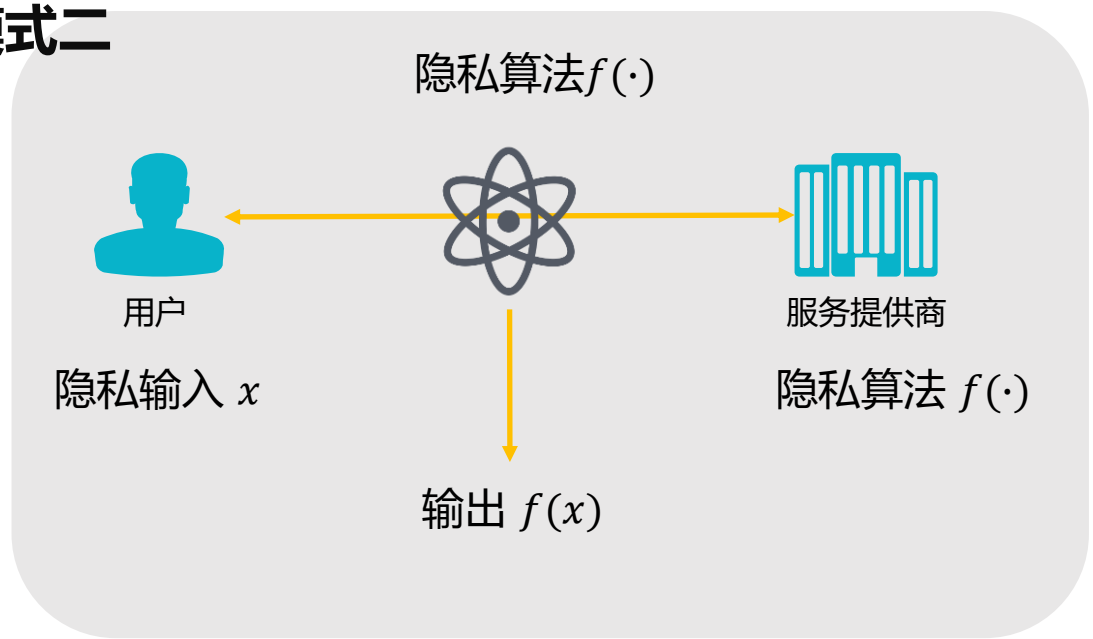
安全多方计算

- 由图灵奖获得者、中国科学院院士姚期智先生于1986年提出
- 多个参与方在不泄露隐私输入的前提下，进行协同计算

模式一



模式二



- 输出结果可按需给到其中一方或者双方共享
- 两种模式本质上是一致的

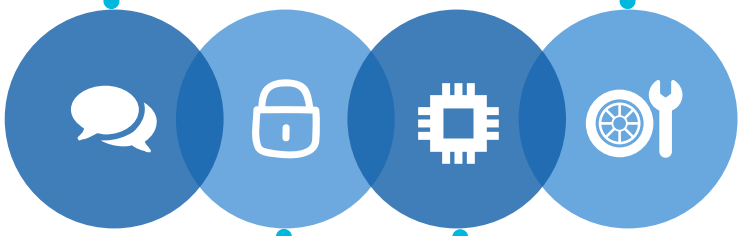
安全多方计算的分类

参与方数量

- 两方
  - 仅由两方参与计算
- 多方
  - 涉及三方及以上的人数共同执行计算过程

故障节点数量

- 大多数诚实节点
  - 大多数参与方都正常执行协议
- 大多数非诚实节点
  - 大部分节点以任意方式破坏协议执行过程



安全模型

- 半诚实敌手
  - 严格遵守协议来执行
  - 不会发送恶意消息
  - 根据己方执行所产生的中间数据尝试获取对方的隐私数据
- 恶意敌手
  - 以任意方式破坏协议正常执行
  - 计算过程中，参与方强行退出
  - 参与方输入错误的原始数据

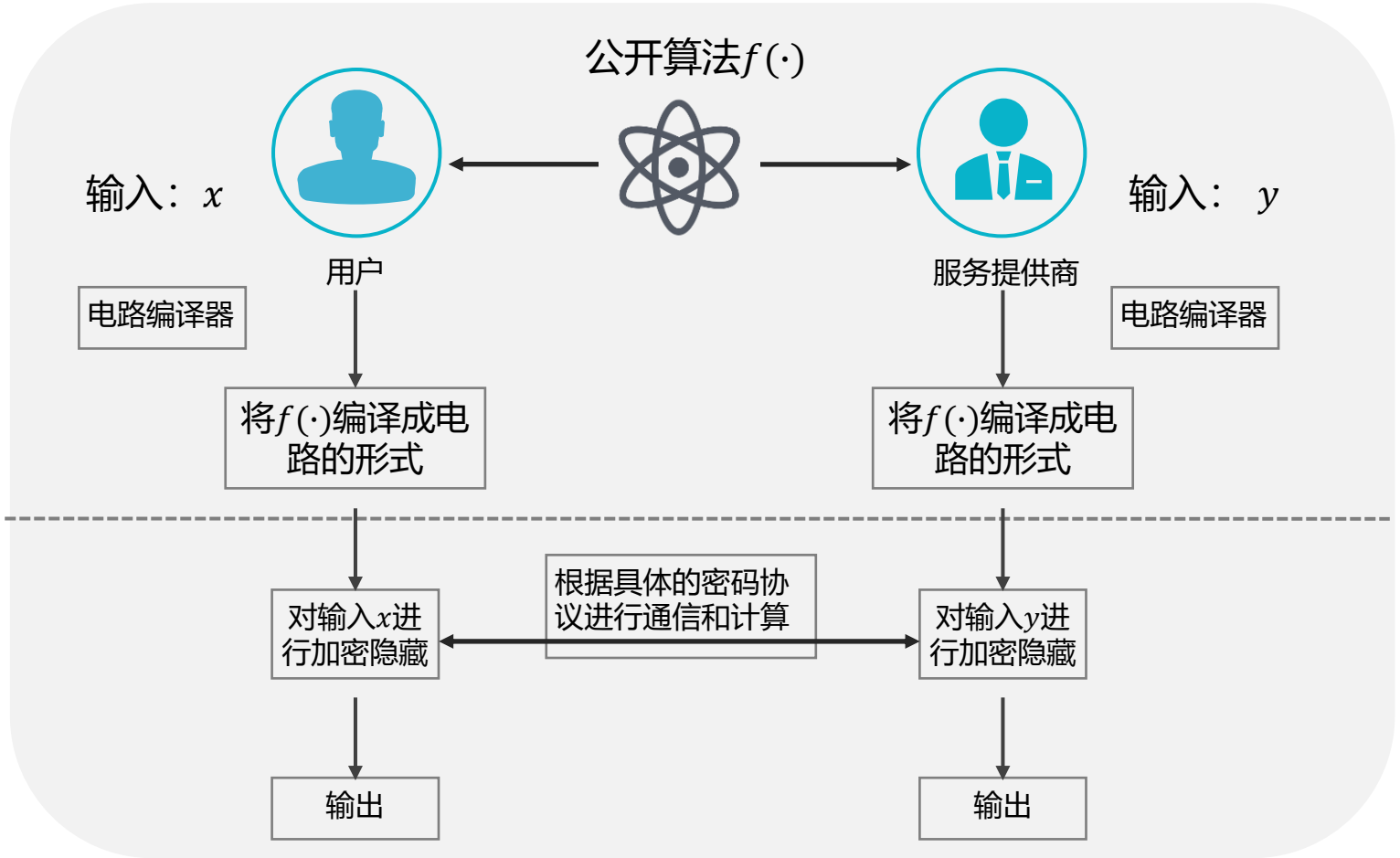
计算模型

- 布尔电路
  - 计算逻辑由AND、XOR等逻辑门构成
- 算术电路
  - 计算逻辑由加法、乘法等算术门构成

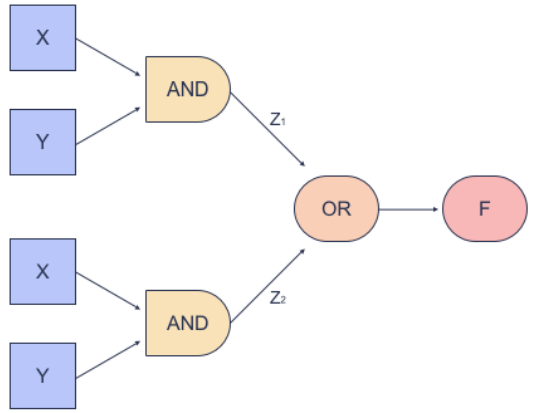
在密码学协议设计中，为了证明其安全性，首先需要定义出其明确的安全性需求。我们无法保证一个协议在任何条件下都是安全的，只有当敌手的行为定义明确，才能给出此敌手模型下所设计协议相应的安全性证明。

两方计算的基本流程

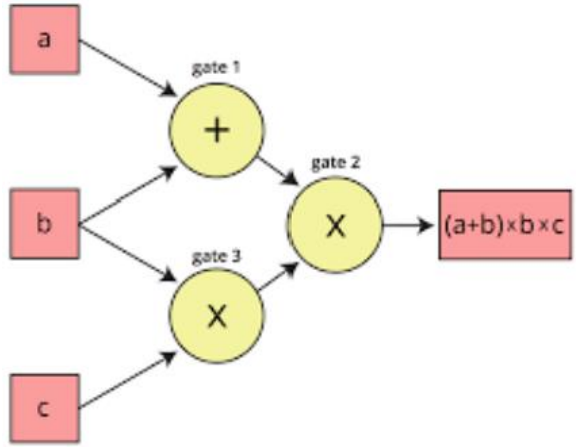
通用算法的基本流程：



布尔电路



算术电路

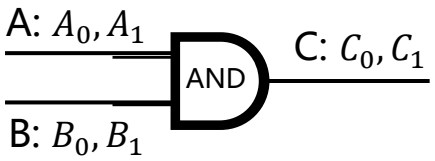


相关原理：GC+OT（布尔电路）

如何获得混淆电路

1. 任意计算逻辑均可转换为布尔电路，由计算双方中的任意一方执行

A、B、C为128比特的随机数



2. 每个门转换为真值表

A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

3. 每条输入线与输出线选择两个随机数替换真值表中的0、1，并用输入线的两个标签加密输出线的标签

A	B	C	Garbled Table
A_0	B_0	C_0	$AES_{A_0,B_0}(C_0)$
A_0	B_1	C_0	$AES_{A_0,B_1}(C_0)$
A_1	B_0	C_0	$AES_{A_1,B_0}(C_0)$
A_1	B_1	C_1	$AES_{A_1,B_1}(C_1)$

由双方任意一方完成

由双方其中一方完成

Boolean Circuit

Truth Table

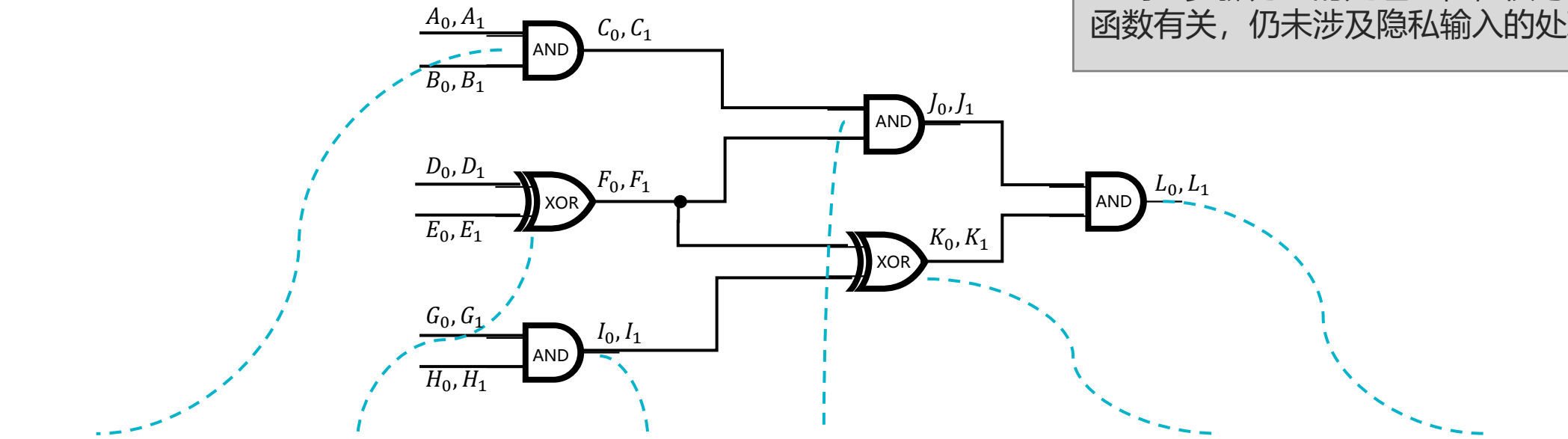
Garbled Table

相关原理：GC+OT（布尔电路）

如何获得混淆电路

4. 逐个门进行混淆，最终得到一个混淆电路

直到此步骤为止的处理过程，仅与计算函数有关，仍未涉及隐私输入的处理。



Garbled Table
$AES_{A_0,B_0}(C_0)$
$AES_{A_0,B_1}(C_0)$
$AES_{A_1,B_0}(C_0)$
$AES_{A_1,B_1}(C_1)$

Garbled Table
$AES_{D_0,E_0}(F_0)$
$AES_{D_0,E_1}(F_1)$
$AES_{D_1,E_0}(F_1)$
$AES_{D_1,E_1}(F_0)$

Garbled Table
$AES_{G_0,H_0}(I_0)$
$AES_{G_0,H_1}(I_0)$
$AES_{G_1,H_0}(I_0)$
$AES_{G_1,H_1}(I_1)$

Garbled Table
$AES_{C_0,F_0}(J_0)$
$AES_{C_0,F_1}(J_0)$
$AES_{C_1,F_0}(J_0)$
$AES_{C_1,F_1}(J_1)$

Garbled Table
$AES_{F_0,I_0}(K_0)$
$AES_{F_0,I_1}(K_1)$
$AES_{F_1,I_0}(K_1)$
$AES_{F_1,I_1}(K_0)$

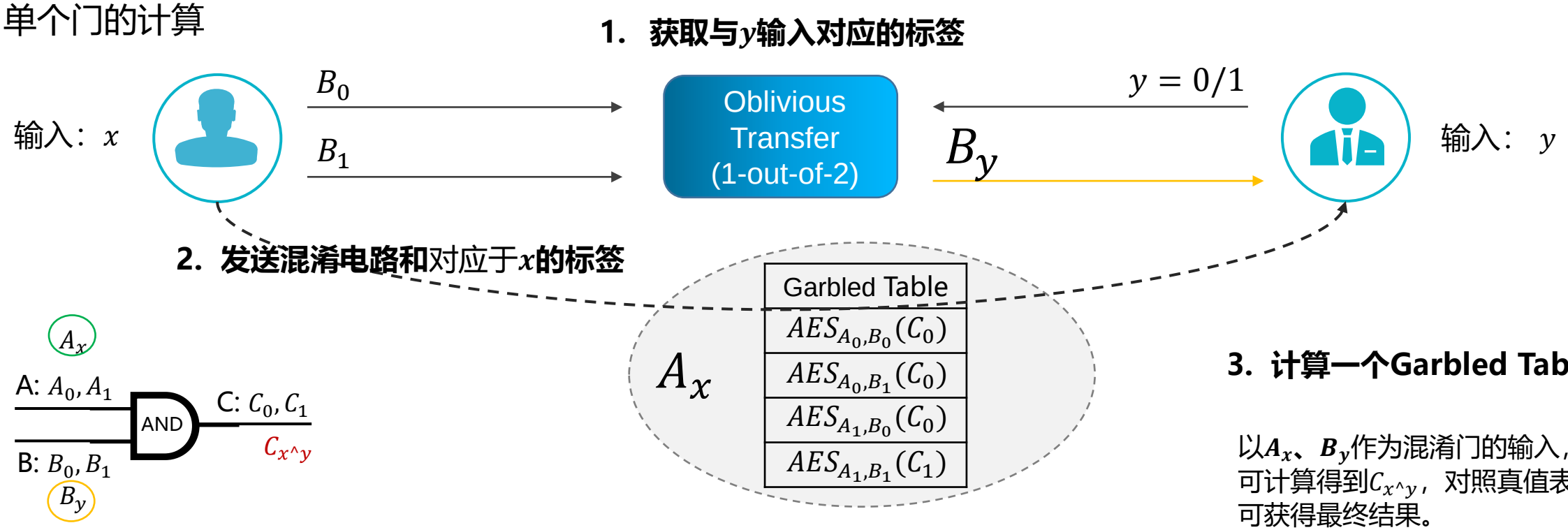
Garbled Table
$AES_{J_0,K_0}(L_0)$
$AES_{J_0,K_1}(L_0)$
$AES_{J_1,K_0}(L_0)$
$AES_{J_1,K_1}(L_1)$

Garbled Circuit

相关原理：GC+OT（布尔电路）

如何计算出混淆电路

单个门的计算



3. 计算一个Garbled Table

以A_x、B_y作为混淆门的输入，则可计算得到$C_{x\wedge y}$，对照真值表即可获得最终结果。

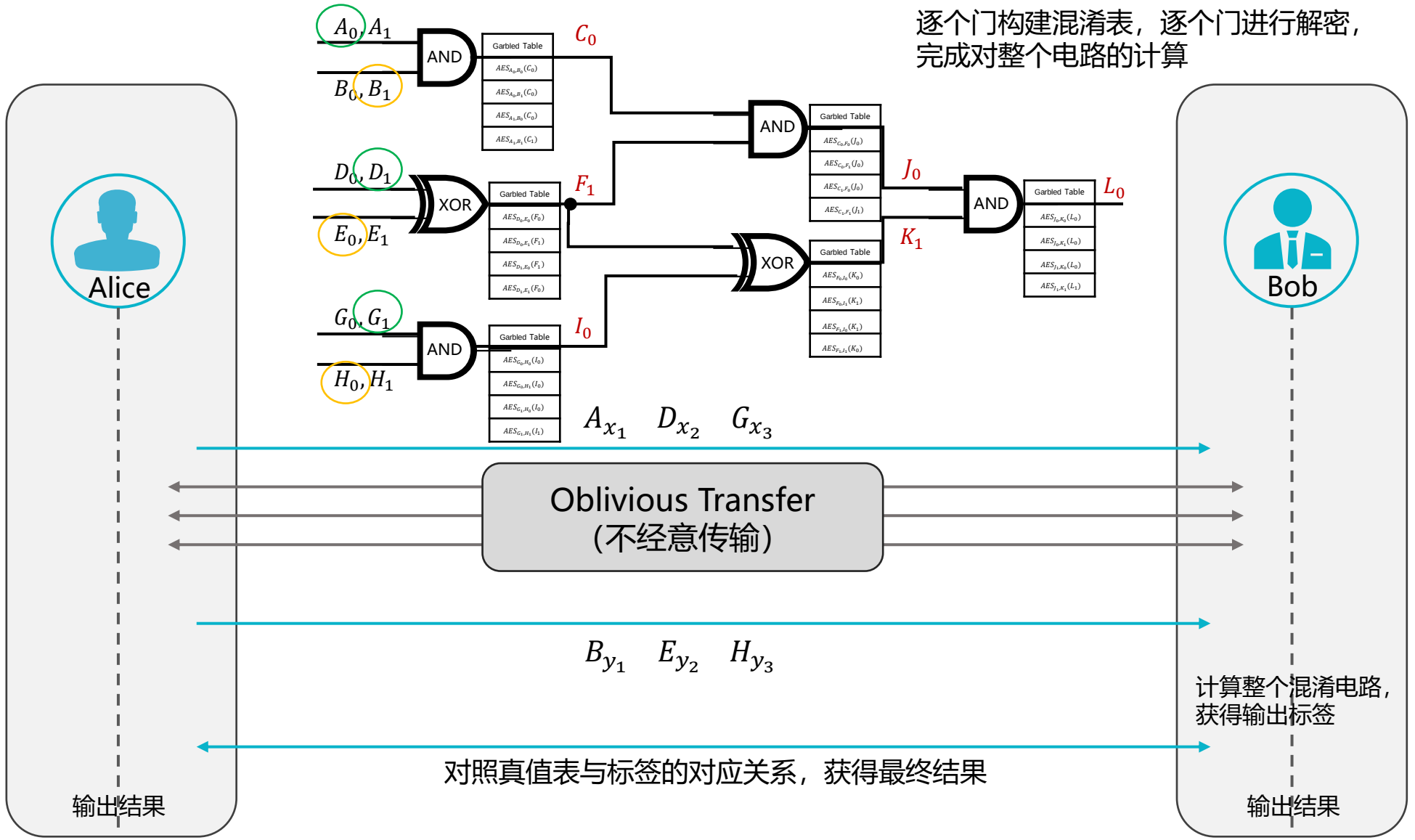
- 对于Alice来说，不知道y具体对应于哪一个
- 整个过程，双方的隐私数据均未离开过本地，全周期数据都是以密文形式在计算

相关原理：GC+OT（布尔电路）

如何计算出混淆电路

发送其中一个标签

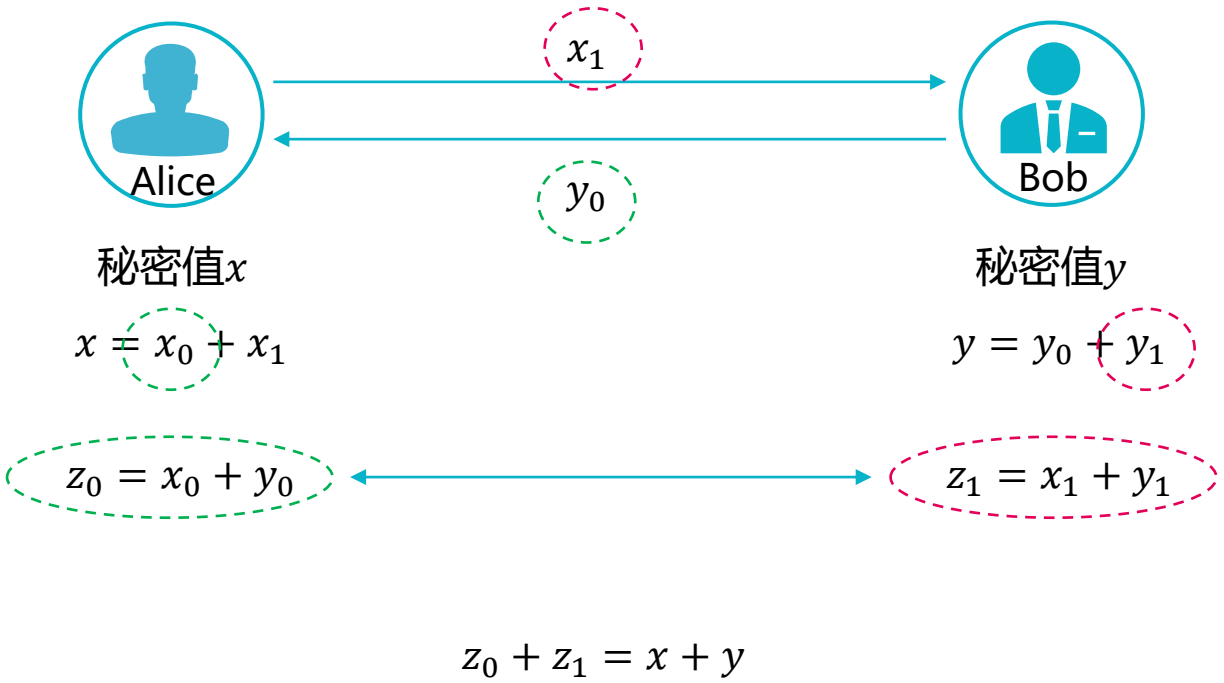
通过OT获取其中一个标签



相关原理：SS+OT（算术电路）

秘密分享，Secret sharing（SS）：计算的中间状态总是分成两份（多份）分享在两个主体（多个主体）之间

秘密值相加



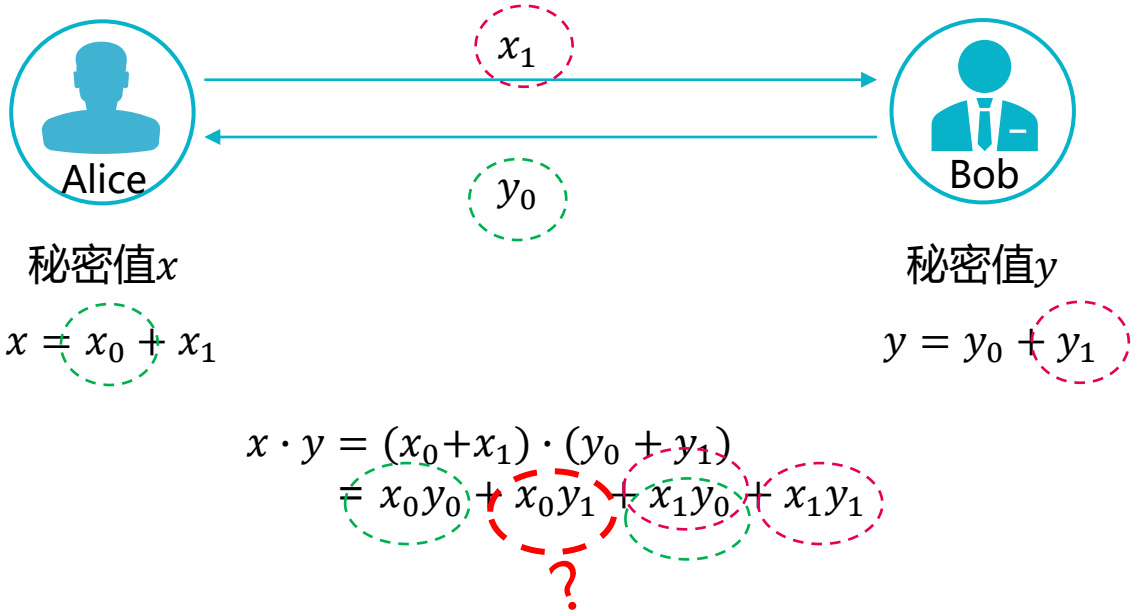
$$z_0 + z_1 = x + y$$

解决过程：

1. Alice与Bob要在不泄露各自秘密值的前提下，计算出两个值之和；
2. 双方分别将各自的秘密值在本地拆分成两份；
3. 双方保留其中一份，将其中一份发送给对方；
4. 对交换后的拆分秘密值之和进一步求和，即可得两个秘密值之和。

相关原理：SS+OT（算术电路）

秘密值相乘

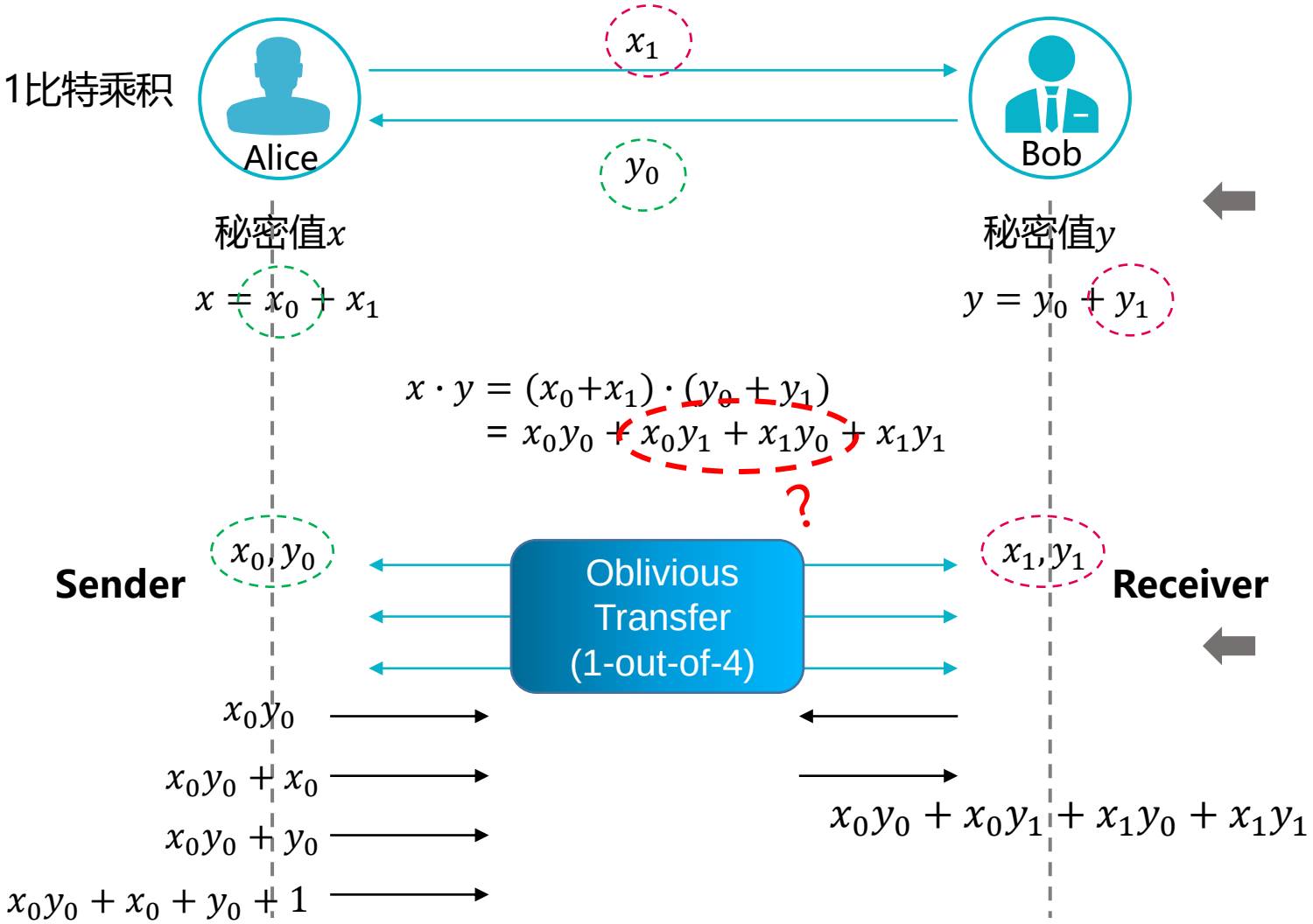


解决过程：

1. Alice与Bob要在不泄露各自秘密值的前提下，计算出两个值的乘积；
2. 双方分别将各自的秘密值在本地拆分成两份；
3. 双方保留其中一份，将其中一份发送给对方；
4. 交换后Alice可以计算出x_0y_0和x_1y_0，Bob能够计算出x_1y_1和x_1y_0，然而Alice和Bob却都无法计算x_0y_1，即如何计算$x_iy_j + x_jy_i$ $(i \neq j)$?

相关原理：SS+OT（算术电路）

秘密值相乘



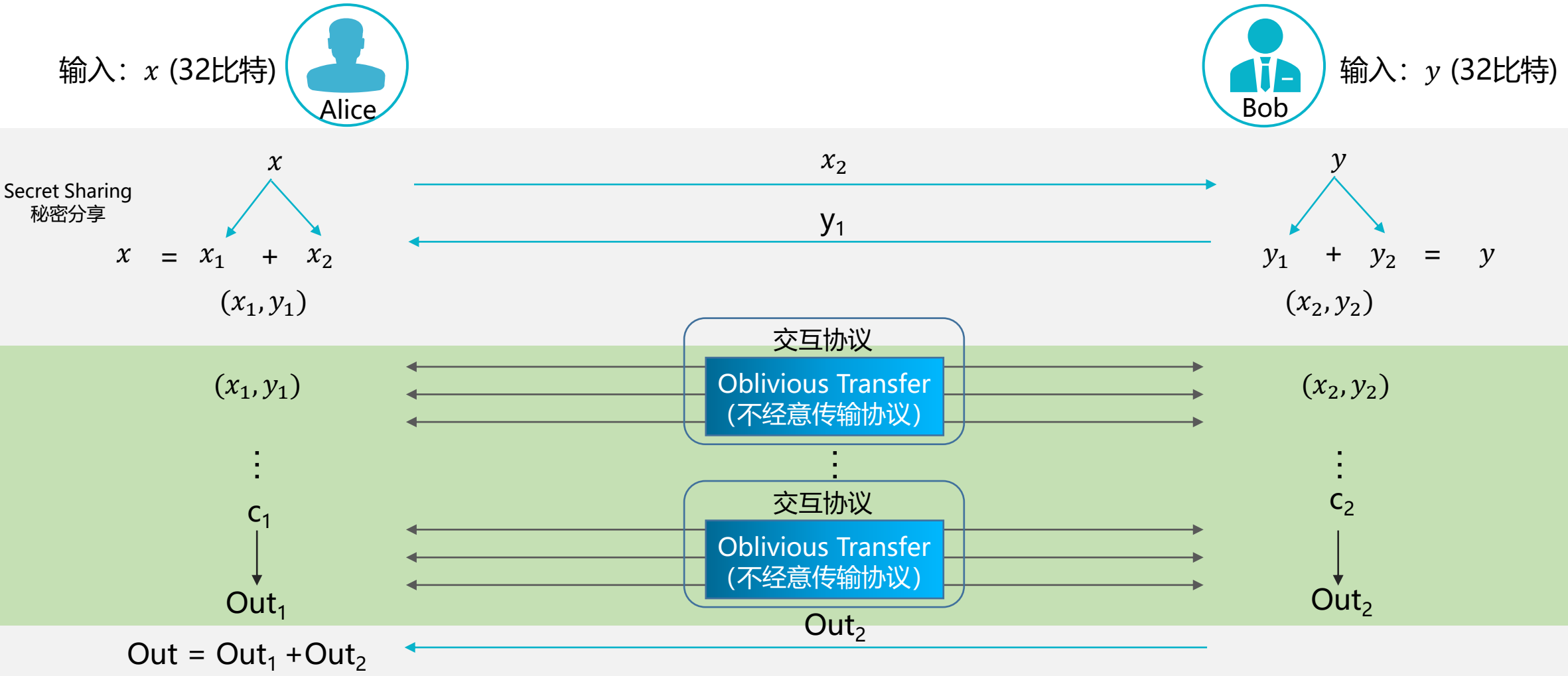
问题：Alice持有x_0, y_0，Bob持有x_1, y_1，如何计算出 $x_0y_0 + x_0y_1 + x_1y_0 + x_1y_1$?

解决过程：Alice作为1-of-4 OT的Sender，构建出一张对应表。

Sender	Receiver
x_0y_0	$x_1 = 0, y_1 = 0$
$x_0y_0 + x_0$	$x_1 = 0, y_1 = 1$
$x_0y_0 + y_0$	$x_1 = 1, y_1 = 0$
$x_0y_0 + x_0 + y_0 + 1$	$x_1 = 1, y_1 =1$

Bob与Alice执行完OT后，即可得到乘积结果。

相关原理：SS+OT（算术电路）



- 整个计算过程，双方的隐私数据均未离开过本地，并且隐私数据在交互过程中从未以任何形式完整出现过。

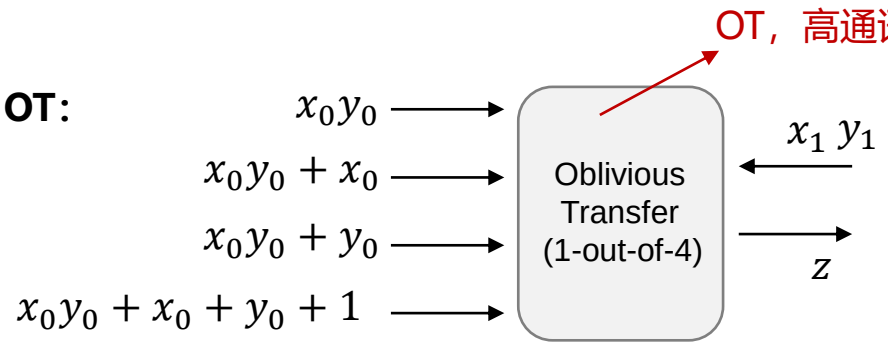
相关原理：Beaver乘法三元组（multiplication triple）

SS+OT中的乘法

- **输入进行秘密共享：**

$$x = x_0 + x_1$$
$$y = y_0 + y_1$$

- **1-out-of 4 OT：**



Beaver triple

通过preprocessing，预先生成一些乘法三元组来减少通讯开销

三元组与输入和电路均无关，因此可在电路计算之前进行

- **Preprocessing: triple generation**

生成三元组(a, b, c)：a、b为随机数，$c = a \cdot b$。秘密共享后，获得$a = a_0 + a_1$、$b = b_0 + b_1$、$c = c_0 + c_1$。每一参与方获得其中一个份额，即参与方P_i 此时有$(x_i, y_i, a_i, b_i, c_i)$。

- **Online computing multiplication**

参与方本地分别计算出：$\varepsilon_i = x_i - a_i$和$\delta_i = y_i - b_i$，则共同可计算出$\varepsilon = x - a$，$\delta = y - b$。而：

$$\begin{aligned} x \cdot y &= (x - a + a) \cdot (y - b + b) \\ &= (x - a) \cdot (y - b) + (y - b) \cdot a + (x - a) \cdot b + a \cdot b \\ &= \varepsilon \cdot \delta + \delta \cdot a + \varepsilon \cdot b + c \end{aligned}$$

每一方都已计算获得

而每一方都已持有(a_i, b_i, c_i)，则均可各自计算出$x \cdot y$ 的秘密共享份额：

$$x_i \cdot y_i = \varepsilon \cdot \delta + \delta \cdot a_i + \varepsilon \cdot b_i + c_i$$

Beaver乘法三元组（multiplication triple）

如何生成三元组

#1 AHE-based Paillier加法同态

(sk, pk)

$$a = a_0 + a_1$$
$$b = b_0 + b_1$$
$$c = c_0 + c_1 = a \cdot b$$

① 选取两个随机数a_0，b_0；
生成同态加密密钥对(sk, pk)

① 选取两个随机数a_1，b_1

② 发送pk，$Enc_{pk}(a_0)$，$Enc_{pk}(b_0)$

③ 选取随机数r，并计算出
$c_1 = a_1 \cdot b_1 - r$，$Enc_{pk}(r)$

④ 同态计算出：
$Enc_{pk}(a_0 \cdot b_1) = b_1 \cdot Enc_{pk}(a_0)$
$Enc_{pk}(b_0 \cdot a_1) = a_1 \cdot Enc_{pk}(b_0)$

通常加上ZKP确保各方发送的密文正确

⑤ 令$v = a_0 \cdot b_1 + b_0 \cdot a_1 + r$，则
$Enc_{pk}(v) = Enc_{pk}(a_0 \cdot b_1 + b_0 \cdot a_1 + r)$
$= Enc_{pk}(a_0 \cdot b_1) + Enc_{pk}(b_0 \cdot a_1) + Enc_{pk}(r)$

⑥ 发送$Enc_{pk}(v)$

⑦ 解密得到$v = Dec_{sk}(Enc_{pk}(v))$，计算出$c_0 = a_0 \cdot b_0 + v$

得到：$c = c_0 + c_1 = a_0 \cdot b_0 + v + a_1 \cdot b_1 - r$
$= a_0 \cdot b_0 + a_0 \cdot b_1 + b_0 \cdot a_1 + a_1 \cdot b_1$
$= (a_0 + a_1) \cdot (b_0 + b_1)$
$= a \cdot b$

Beaver乘法三元组（multiplication triple）

如何生成三元组

#2 OT-based　不经意传输

$$a = a_0 + a_1$$
$$b = b_0 + b_1$$
$$c = c_0 + c_1 = a \cdot b$$

① 选取两个随机数a_0，b_0

② 选取一个随机比特r_0，
构建出OT的输入：
$(r_0, a_0 \oplus r_0)$，

① 选取两个随机数a_1，b_1

③ 1st OT

r_0
$a_0 \oplus r_0$
Oblivious Transfer (1-out-of-2)
b_1

④ 若$b_1 = 0$，则输出为r_0；
若$b_1 = 1$，则输出为$a_0 \oplus r_0$，
则OT结束，获得$a_0 b_1 \oplus r_0$。

⑤ 选取一个随机比特r_1，
构建出OT的输入：
$(r_1, a_1 \oplus r_1)$，

⑥ 2rd OT

b_0
Oblivious Transfer (1-out-of-2)
r_1
$a_1 \oplus r_1$

⑦ 若$b_0 = 0$，则输出为r_1；
若$b_0 = 1$，则输出为$a_1 \oplus r_1$，
则OT结束，获得$a_1 b_0 \oplus r_1$。

⑧ 计算出：$c_0 = r_0 \oplus a_0 b_0 \oplus a_1 b_0 \oplus r_1$

⑧ 计算出：$c_1 = r_1 \oplus a_1 b_1 \oplus a_0 b_1 \oplus r_0$

计算：$c = c_0 \oplus c_1 = r_0 \oplus a_0 b_0 \oplus a_1 b_0 \oplus r_1 \oplus r_1 \oplus a_1 b_1 \oplus a_0 b_1 \oplus r_0$
$= a_0 b_0 \oplus a_1 b_0 \oplus a_1 b_1 \oplus a_0 b_1 = (a_0 \oplus b_0)(a_1 \oplus b_1) = ab$

相关原理：OT算法（Oblivious Transfer）

OT算法的构造通常是基于公钥加密

m_0

m_1

发送者

$b = 0/1$

接收者

② 发送pk_0和pk_1

① 生成公钥对$(sk_b, pk_b) \leftarrow Gen(1^k)$
和另一个随机数$pk_{1-b} \leftarrow rand()$

③ 使用两个公钥分别加密两个数据：
$c_0 \leftarrow Enc_{pk_0}(m_0)$
$c_1 \leftarrow Enc_{pk_1}(m_1)$

④ 发送c_0和c_1

⑤ 使用私钥 sk_b解密密文：
$m_b \leftarrow Dec_{sk_b}(c_b)$

如果不是仅生成一个随机数，而是公私钥对，怎么办？

无法获知 b

无法获知 m_{1-b}

Naor-Pinkas OT构造方案

随机预言机安全模型

G是一个阶为q的群，g为其生成元， H为随机预言机

m_0

m_1

发送者

$b = 0/1$

接收者

① 随机选取群元素$C \in Z_q$

② 发送C

③ 生成公钥对 $(sk_b = k \in Z_q, pk_b = g^k)$

④ 计算另一个公钥 $pk_{1-b} = C/pk_b$

⑤ 发送pk_0和pk_1

⑥ 选取两个随机数$r_0 \in Z_q$和$r_1 \in Z_q$，使用ElGamal分别加密两个数据：

$c_0 =< g^{r_0}, H(pk_0^{r_0}) \oplus m_0 >$

$c_1 =< g^{r_1}, H(pk_1^{r_1}) \oplus m_1 >$

⑦ 发送c_0和c_1

⑧ 使用私钥 sk_b 计算 $H\left((g^{r_b})^{sk_b}\right) = H(pk_b^{r_b})$，即可得到：

$m_b = H(pk_b^{r_b}) \oplus c_b$

无法获知 b

无法获知 m_{1-b}

Naor-Pinkas OT构造方案

标准安全模型

G是一个阶为q的群，g为其生成元

m_0

m_1

发送者

$\sigma = 0/1$

接收者

② 发送 $(g, g^a, g^b, g^{c_0}, g^{c_1})$

① 选取$a, b \leftarrow Z_q$
计算出$c_\sigma = ab$，并
选取$c_{1-\sigma} \leftarrow Z_q$

③ 验证$g^{c_0} \neq g^{c_1}$
并生成随机数$(r_0, s_0) \leftarrow Z_q$
和$(r_1, s_1) \leftarrow Z_q$

④ 计算出
$w_0 = (g^a)^{s_0} \cdot g^{r_0}$、$w_1 = (g^a)^{s_1} \cdot g^{r_1}$
以及
$k_0 = (g^{c_0})^{s_0} \cdot (g^b)^{r_0}$、$k_1 = (g^{c_1})^{s_1} \cdot (g^b)^{r_1}$

⑥ 发送 $(w_0, w_1, Enc(k_0, m_0), Enc(k_1, m_1))$

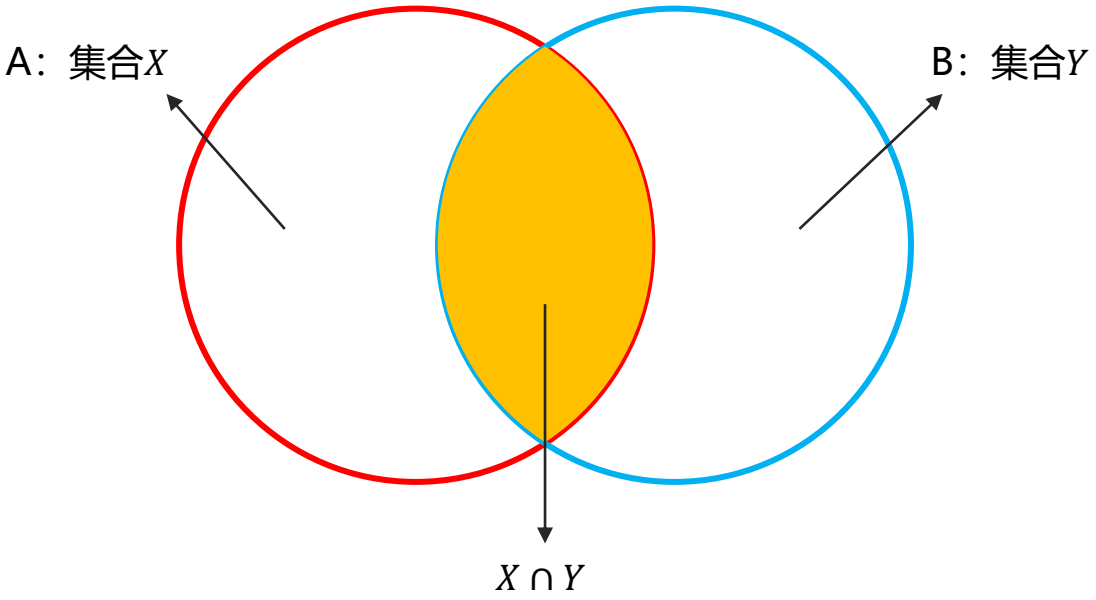
⑦ 计算 $(w_\sigma)^b = (g^{ab})^{s_\sigma} \cdot (g^b)^{r_\sigma}$，$k_\sigma$
然后解密$Enc\left((g^{c_\sigma})^{s_\sigma} \cdot (g^b)^{r_\sigma}, m_\sigma\right)$

k_σ → m_σ

⑤ 使用k_0和k_1采用ElGamal分别加密
m_0和m_1

相关原理：PSI（private set intersection）

PSI，隐私集合交集，是安全计算中的一种特定协议，用于计算出多个隐私集合的交集。可在许多应用场景下发挥作用，比如共享名单查询、场景匹配、企业信息共享等。



输入： $X = \{x_1, x_2, \dots, x_n\}$ $Y = \{y_1, y_2, \dots, y_m\}$

输出： $X \cap Y$ $X \cap Y$

- 双方无法获得除结果之外对方输入中的其它隐私信息
- 输出可根据实际情况，选择双方同时获得或者其中一方获得

PSI方案构造

基于哈希的PSI方案


输入： $X = \{x_1, x_2, \dots, x_n\}$ $Y = \{y_1, y_2, \dots, y_m\}$

$\{H(x_1), H(x_2), \dots, H(x_n)\}$ $\{H(y_1), H(y_2), \dots, H(y_m)\}$

$\{H(y_1), H(y_2), \dots, H(y_m)\}$ ←

比较哈希值 $H(x_i) \stackrel{?}{=} H(y_j)$

- 实现快
- 但一旦输入空间不大，方案会变得不安全

基于DH的PSI方案


输入： $X = \{x_1, x_2, \dots, x_n\}$ $Y = \{y_1, y_2, \dots, y_m\}$

① 选取随机数a ① 选取随机数b

② 计算出：$\{H(x_1)^a, H(x_2)^a, \dots, H(x_n)^a\}$ ② 计算出：$\{H(y_1)^b, H(y_2)^b, \dots, H(y_m)^b\}$

→ ←

③ 计算：$\{(H(y_1)^b)^a, (H(y_2)^b)^a, \dots, (H(y_m)^b)^a\}$ ③ 计算：$\{(H(x_1)^a)^b, (H(x_2)^a)^b, \dots, (H(x_n)^a)^b\}$

比较第③步各自计算结果

PSI方案构造

基于OT的PSI方案

$\binom{2}{1}$-OT　　**比较两个长度为σ比特的元素x是否等于y**


P1


P2

输入： $X = \{x_1, x_2, \dots, x_m\}$　　$Y = \{y_1, y_2, \dots, y_n\}$

比较$x_i \triangleq y_j$

判断：$x_i \in Y = \{y_1, y_2, \dots, y_n\}$

确定：$X \cap Y$


P1


P2

$\boldsymbol{x = x[\sigma - 1] \dots x[1]x[0]}$　　$\boldsymbol{y = y[\sigma - 1] \dots y[1]y[0]}$

① 选取σ对长度为l比特的随机串(s_0^i, s_1^i)，$1 \le i \le \sigma$

$\binom{2}{1}$-OT_l^σ

② 输出$s_{y[i]}^i$，$1 \le i \le \sigma$

③ 计算$m_1 = \oplus_{i=1}^{\sigma} s_{x[i]}^i$

③ 计算$m_2 = \oplus_{i=1}^{\sigma} s_{y[i]}^i$

④ 发送m_1

⑤ 比较$m_1 \stackrel{?}{=} m_2$

[1] Pinkas-Schneider (2014). *Faster Private Set Intersection based on OT extension.*

PSI方案构造

基于OT的PSI方案


P1


P2

输入： $X=\{x_1,x_2,\dots,x_m\}$ $Y=\{y_1,y_2,\dots,y_n\}$

比较$x_i \triangleq y_j$

判断：$x_i \in Y=\{y_1,y_2,\dots,y_n\}$

确定：$X \cap Y$

比较长度为σ比特的元素x是否与集合$Y=\{y_1,y_1,\dots y_n\}$中的任意元素相等

$x=x[\sigma-1]\dots x[1]x[0]$

$$y=\begin{bmatrix} y_1[\sigma-1]\dots y_1[1]y_1[0] \\ y_2[\sigma-1]\dots y_2[1]y_2[0] \\ \dots \\ y_n[\sigma-1]\dots y_n[1]y_n[0] \end{bmatrix}$$

① 按比特切分为t块

$x=x[t-1]||\dots||x[1]||x[0]$

① 按比特切分为t块

$$y=\begin{bmatrix} y_1[t-1]||\dots||y_1[1]||y_1[0] \\ y_2[t-1]||\dots||y_2[1]||y_2[0] \\ \dots \\ y_n[t-1]||\dots||y_n[1]||y_n[0] \end{bmatrix}$$

② 选取t组元素个数为N，元素长度为l比特的随机串$(s_0^i,s_1^i,\dots,s_{N-1}^i)$，$1\le i\le t$

$N=2^{\sigma/t}$

$x[i]\in[0,N-1]$ → $\binom{N}{1}$-OT_{nl}^t ← $(s_0^i,s_1^i,\dots,s_{N-1}^i)$

$s_{x[i]}^i$

③ 输出$s_{x[i]}^i$，$1\le i\le t$

④ 计算$m_1=\oplus s_{x[i]}^i$，$1\le i\le t$

④ 计算 $m_{2,j}=\oplus s_{y_j[i]}^i$，$1\le i\le t, 1\le j\le n$

⑤ 发送$m_{2,j}$，$1\le j\le n$ ←

⑥ 比较$m_1 \stackrel{?}{=} m_{2,j}$，$1\le j\le n$

2 同态加密

什么是同态加密

- 1978年，Rivest, Adleman, Dertouzos提出同态加密的概念；Craig Gentry进一步在2009年给出第一个全同态方案
- 可以在密文状态下进行计算
- 计算函数 f 的不同，可分为部分同态（加法同态，乘法同态）和全同态（任何操作）

密码学中的圣杯

同态加密：构造方案

ElGamal同态加密

① 随机选取$x \in Z_p$，
并计算出$y = g^x \ (mod\ p)$，
私钥为x，公钥为(y, g, p)

② 发送公钥(y, g, p)

③ 加密消息M，首先随机选取k
$\in Z_p$，计算出：
$a = g^k \ (mod\ p)$，
$b = y^k M \ (mod\ p)$，
即得到加密后的数据密文(a, b)

④ 发送密文(a, b)

⑤ 使用私钥x解密可获得明文
消息：$M = \frac{b}{a^x} \ (mod\ p)$

对于消息M_1，获得密文(a_1, b_1)，
$a_1 = g^{r_1} \ (mod\ p)$，
$b_1 = y^{r_1} M_1 \ (mod\ p)$；
对于消息M_2，获得密文(a_2, b_2)，
$a_2 = g^{r_2} \ (mod\ p)$，
$b_2 = y^{r_2} M_2 \ (mod\ p)$。
直接对两个密文进行相乘，并进行解密，我们得到：

$$\frac{b_1 b_2}{{a_1}^x {a_2}^x} = M_1 M_2 (mod\ p)$$

具备同态乘法性质，如何使其满足同态加法？

同态加密：构造方案

ElGamal同态加密

① 随机选取$x \in Z_p$，
并计算出$y = g^x \ (mod\ p)$，
私钥为x，公钥为(y, g, p)

② 发送公钥(y, g, p)

③ 加密消息M，首先随机选取$k \in Z_p$，计算出：
$a = g^k \ (mod\ p)$，
$b = y^k g^M \ (mod\ p)$，
即得到加密后的数据密文(a, b)

④ 发送密文(a, b)

⑤ 使用私钥x解密可获得明文消息：$M = \frac{b}{a^x} \ (mod\ p)$

加密方案的小修改

对于消息M_1，获得密文(a_1, b_1)，
$a_1 = g^{r_1} \ (mod\ p)$，
$b_1 = y^{r_1} g^{M_1} \ (mod\ p)$；
对于消息M_2，获得密文(a_2, b_2)，
$a_2 = g^{r_2} \ (mod\ p)$，
$b_2 = y^{r_2} g^{M_2} \ (mod\ p)$。
直接对两个密文进行相乘，并进行解密，我们得到：

$$\frac{b_1 b_2}{{a_1}^x {a_2}^x} = \frac{y^{r_1} g^{M_1} y^{r_2} g^{M_2}}{(g^{r_1})^x (g^{r_2})^x} = g^{M_1 + M_2} (mod\ p)$$

离散对数求解，即可获得明文数据相加的结果。

然鹅却无法同时满足同态加法和同态乘法性质。

同态加密：构造方案

RSA乘法同态

给定分别对消息m_1和m_2的密文c_1、c_2：

$$c_1 = {m_1}^e \ mod \ N$$

$$c_2 = {m_2}^e \ mod \ N$$

$$\Rightarrow \quad c_1 \cdot c_2 = (m_1 \ \cdot m_2 \ )^e \ mod \ N$$

Paillier加法同态

给定分别对消息m_1和m_2的密文c_1、c_2：

$$c_1 = g^{m_1} \ mod \ N^2$$

$$c_2 = g^{m_2} \ mod \ N^2$$

$$\Rightarrow \quad c_1 \cdot c_2 = (g)^{m_1+m_2} \ mod \ N^2$$

DGHV全同态

给定分别对消息m_1和m_2的密文c_1、c_2：

$$c_1 = p \cdot q_1 + 2r_1 + m_1$$

$$c_2 = p \cdot q_2 + 2r_2 + m_2$$

$$\Rightarrow \quad c_1 + c_2 = p \cdot q' + 2r' + m_1 + m_2$$

$$c_1 \cdot c_2 = p \cdot q'' + 2r'' + m_1 \cdot m_2$$

安全计算算法比较

算法	适用参与方	优点	缺点	适用场景
GC+OT (Yao’ s Protocol)	两方	• 计算速度快 • ~1000w gate/s • 通信次数少	• 通信量大，对带宽要求高	• 带宽大 • 逻辑操作多（比较，MUX，非线性函数等）
SS+OT (GMW Protocol)	两方及以上	• 计算速度快 • ~500w gate/s • 单次的通信量少	• 通信次数多，对时延要求高	• 时延低 • 算术操作多（矩阵乘法，线性函数等）
HE	两方及以上	• 通信次数少 • 计算复杂度不对称	• 计算复杂度高	• 有强大的计算能力（比如云） • 通信要求受限

在特定的场景下，三种方法结合各自的优势进行组装，能有效提高性能

- 输出结果可按需给到其中一方或者双方共享
- 两种模式本质上是一致的

应用：数据密态计算

场景：两机构之间由于政策或者利益考虑，无法彼此公开各自的数据，但存在使用对方数据的需求，以求最大化已方数据的价值

解决方案：两机构各自部署MPC算法，可实现双方数据在不出本地的情况下，进行数据协同分析处理。

1

两方各自在本地完成MPC软件系统的部署

2

两方事先约定所要使用的数据分析算法，并由其中一方将算法编译成计算所要使用的电路文件

3

编译电路文件的一方将此电路文件线下或者在线发送给另一方。电路文件由于不涉及双方的任何隐私数据，因此可以公开

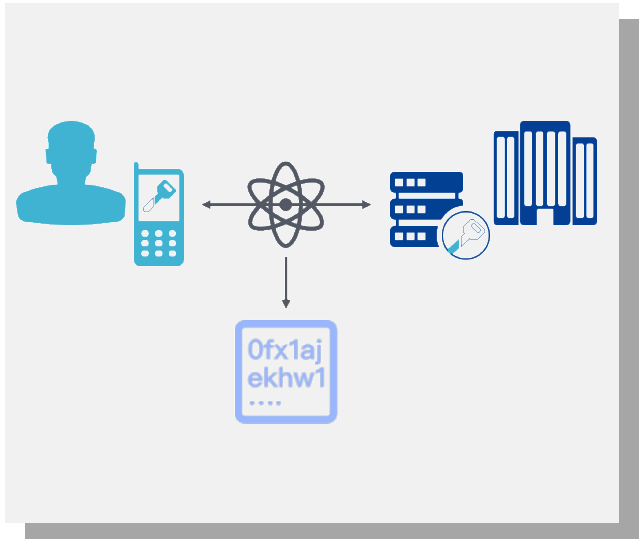
4

双方各自在本地预处理自己的隐私数据，得到MPC计算所需的结构化数据，双方一起完成MPC计算，获得计算结果

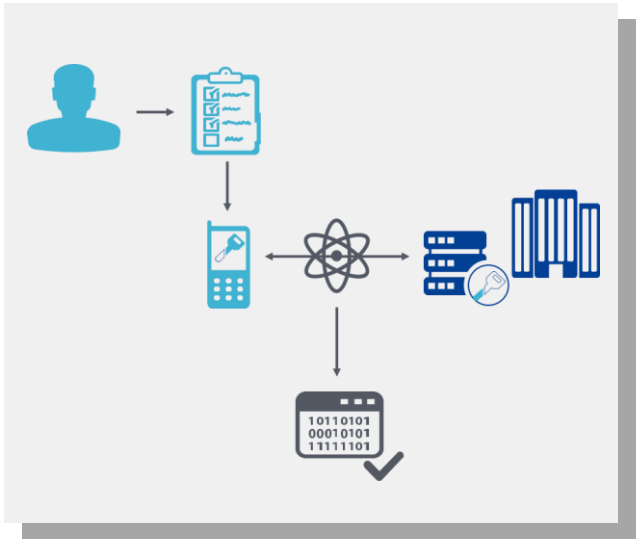
应用：密钥管理

- 门限签名技术是MPC中的一个具体例子，用于实现多方参与的分布式数字签名
- 在整个生命周期中，私钥从未出现，一直被安全地拆分由多个实体分别存储

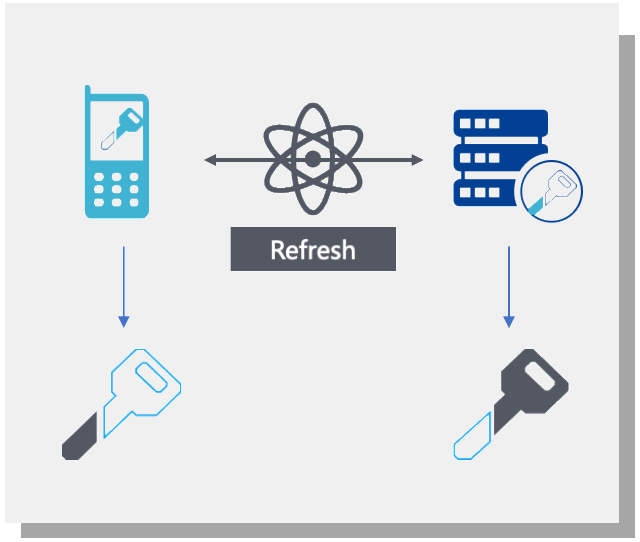
分布式密钥生成，完整私钥从未出现，单一的碎片完全随机



基于MPC，碎片无需聚合，生成标准签名



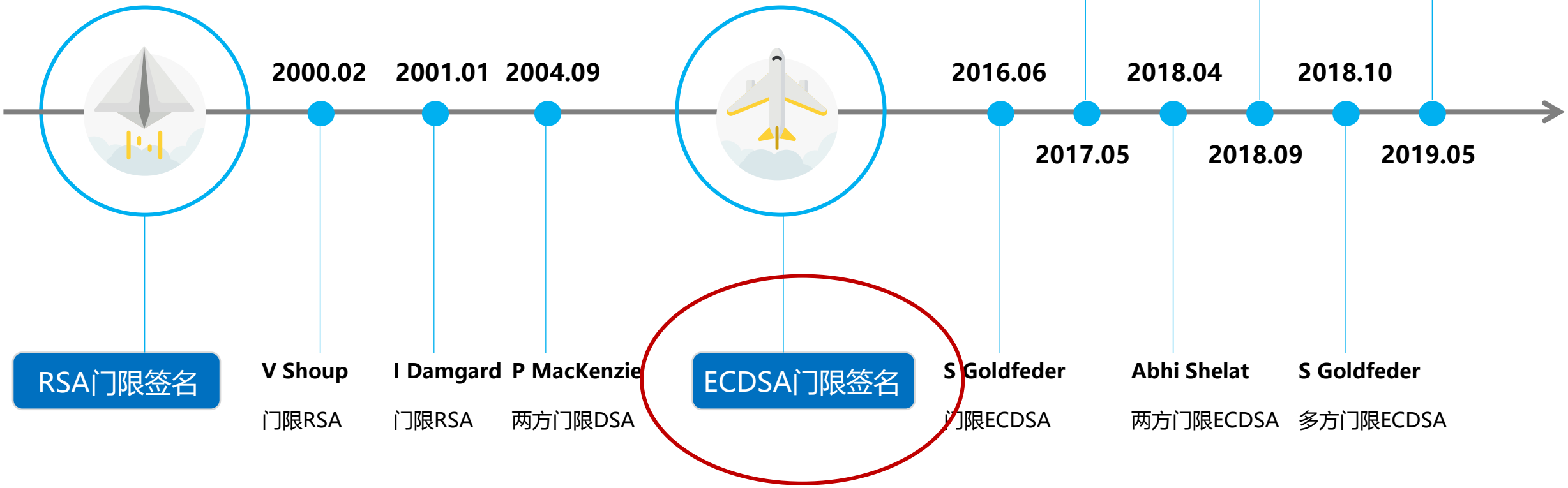
碎片按需刷新，进一步提升安全性



门限签名发展史

为何需要门限签名?

- 保护私钥：私钥丢失意味着对应账户的资产丢失，降低单点失败风险
- 托管需求：可实现多方共同控制一个账户



应用：密钥管理

技术实现：基于RSA的门限签名

标准RSA签名方案

- 选取两个大素数p和q；
- 计算$N = pq$，以及$\varphi(N) = (p-1)(q-1)$；
- 选取另一个整数e，使得$gcd(e, \varphi(N)) = 1$；
- 计算出d，满足$ed = 1\ mod\ \varphi(N)$

公钥(e, n)，私钥(d, n)

签名者对消息m进行签名：$sig = m^d\ mod\ N$

两方RSA签名方案

- 计算出d后，随机生成d_1，计算出$d_2 = d - d_1$；
- A获得密钥碎片d_1， A获得密钥碎片d_2，其中$d_1 + d_2 = d$；
- A计算$sig_1 = m^{d_1}\ mod\ N$，并将sig_1发送给B；
- B计算$sig_2 = m^{d_2}\ mod\ N$，并计算出$sig = sig_1 \cdot sig_2\ mod\ N$；

密钥生成

- B验证$sig^e = (m^{d_1} \cdot m^{d_2})^e = m^{e(d_1+d_2)} = m^{ed} = m^1 = m\ mod\ N$

签名有效!

签名生成

服务器A

服务器B

标准ECDSA签名

W为一个阶为q的椭圆曲线上的群，生成元为G，签名者对消息m进行签名：

KeyGen

1. 选取随机数$x \leftarrow Z_q$，计算$Q = xG$。私钥为x，公钥则为Q；

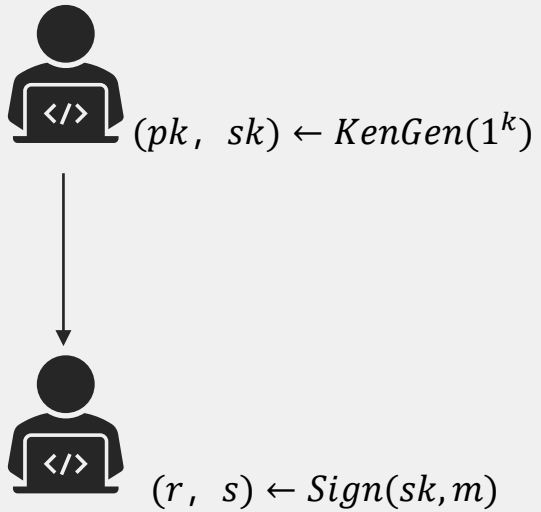
Sign

1. 选取随机数$k \in Z_q^*$；
2. 计算$R \leftarrow kG$，令$R = (R_x, R_y)$；
3. 计算$r = R_x \bmod q$，$s = k^{-1}(m + rx) \bmod q$；
4. 输出签名(r, s)。

kG和k^{-1}

Verification

1. 验证r和s都是区间$[1, q-1]$上的整数；
2. 计算$u = s^{-1} \bmod q$；
3. 计算$v_1 = mu \bmod q$，$v_2 = ru \bmod q$；
4. 计算$X = v_1 G + v_2 Q$，令$X = (X_x, X_y)$；
5. 如果$r = X_x$，则签名有效。



门限签名：预备知识

Threshold Cryptography

参与方总数为n，当至少需要其中t个参与方才能完成对消息的签名，这样的数字签名方案称为(t,n)门限签名方案。

Secret Sharing

秘密信息σ持有者为了将其共享给多个其他参与方（$\sigma \in Z_q$），首先生成Z_q上的一个阶为$(t-1)$的随机多项式$f(\cdot)$，使得$f(0)=\sigma$。通过此多项式计算出提供给每一方的秘密值碎片σ_i，多项式定义为：

$$f(x)=\sigma+a_1x+a_2x^2+\cdots+a_{t-1}\,x^{t-1}\ mod\ q$$

计算出$\sigma_i=f(i)\ mod\ q$，$i\in[1,n]$，并将计算出的σ_i发送给对应的参与方。

如何保证接收者收到的σ_i的正确性?

Verifiable Secret Sharing

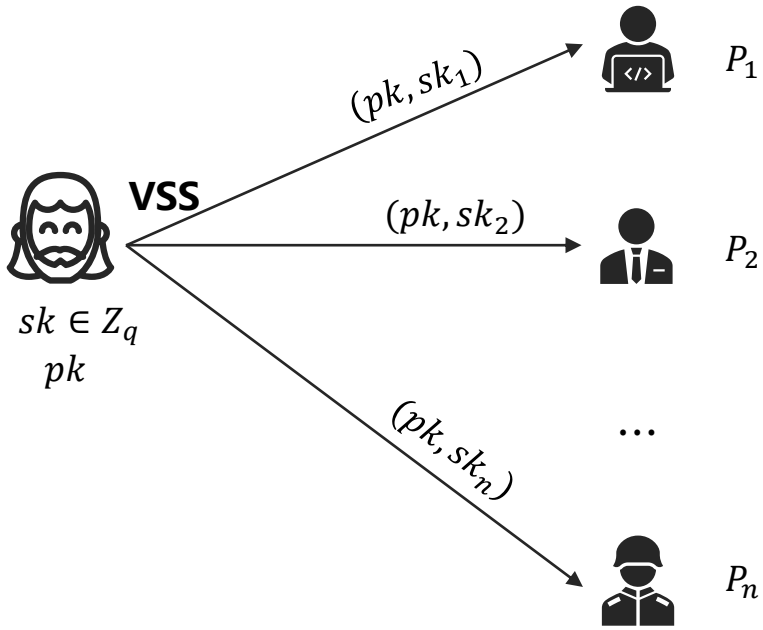
在计算出σ_i后，为了保证接收数据的正确性，即需确保多项式的系数正确。计算出$v_j=g^{a_j}\ mod\ q$，$j\in[1,t]$，这里$v_0=g^{\sigma}\ mod\ q$，并将所有的v_j公开。各方接收到σ_i后，通过下面的计算，即可验证σ_i的正确性：

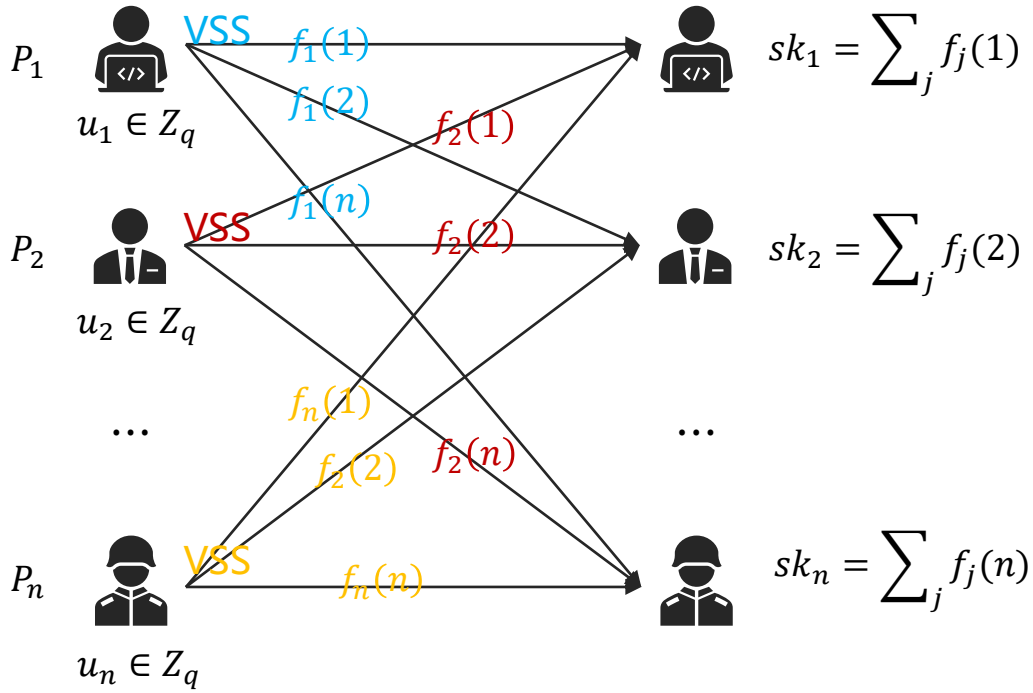
$$v_0{v_1}^{i}{v_2}^{i^2}\cdots{v_{t-1}}^{i^{t-1}}\ (mod\ q)=g^{\sum_{j=0}^{t-1}a_ji^j}=g^{f(i)}=g^{\sigma_i}$$

这是由于：$g^{a_0}(g^{a_1})^{i}(g^{a_2})^{i^2}\cdots(g^{a_{t-1}})^{i^{t-1}}\ (mod\ q)=g^{\sum_{j=0}^{t-1}a_ji^j}\ (mod\ q)=g^{f(i)}=g^{\sigma_i}$。

门限ECDSA签名

密钥生成：Distributed Key Generation





① 可信方选取随机数$sk \in Z_q$；
② 可信方随机选择Z_q上的多项式$f(x) = a_0 + a_1 x + \cdots + a_{t-1} x^{t-1}$，，这里$a_0 = sk$；
③ 计算出$f(i)$，$1 \le i \le n$，并发送给对应的参与方P_i，每一方的私钥碎片则为$sk_i = f(i)$。

① 参与方各自选取随机数$u_i \in Z_q$；
② 每一方P_i随机选择Z_q上的多项式$f_i(x) = a_{i0} + a_{i1} x + \cdots + a_{i(t-1)} x^{t-1}$，$1 \le j \le n$，这里$a_{i0} = u_i$；并计算出$A_{ik} = g^{a_{ik}}$发送给其他所有方，$0 \le k \le t-1$，这里取$y_i = g^{a_{i0}} = g^{u_i}$；
③ 计算出$f_i(j)$，$1 \le j \le n$，并发送给对应的参与方P_j；
④ 全部发送完成后，每一方P_i均能获得$f_1(i)$、$f_2(i)$、…、$f_n(i)$，$1 \le i \le n$。计算出完整公钥$y = \prod_i y_i$；P_i对应的私钥碎片$sk_i = \sum_j f_j(i) \ mod \ q$，$1 \le j \le n$。

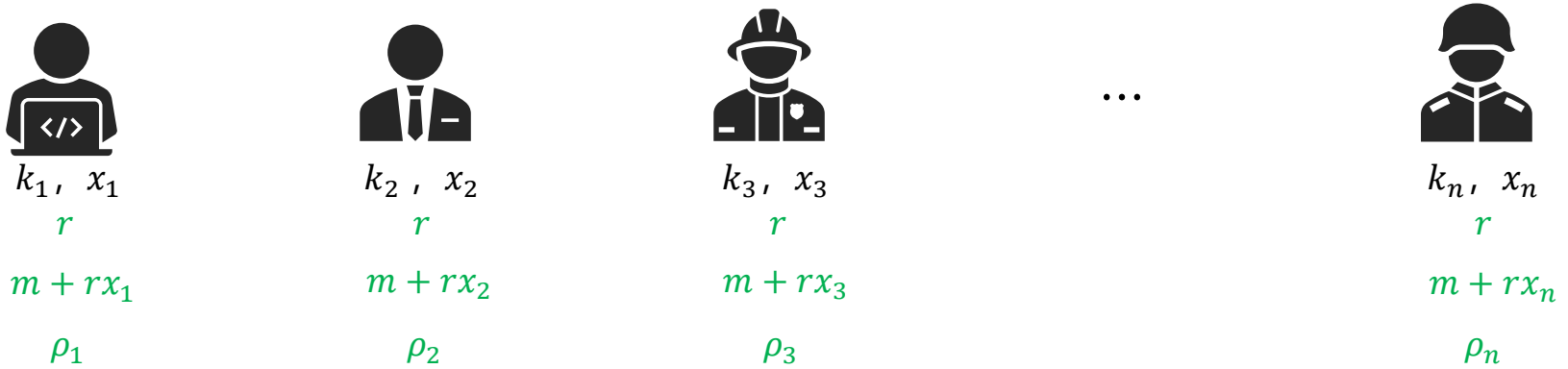
门限ECDSA签名

分布式签名： Distributed Signing

生成的签名与标准签名格式一致

$s = k^{-1}(m + rx) \ mod \ q$

如何计算出 kG和k^{-1}



① 参与各方随机选择k_i，其中$k_1 + k_2 \cdots + k_n = k$；

② 各自本地计算出$k_i \cdot G$，并发送给其他所有方，然后计算：

- ✓ $k \cdot G = k_1 \cdot G + k_2 \cdot G + \cdots + k_n \cdot G$，即获得$r = (k \cdot G)_x$
- ✓ 各自本地计算出$m + rx_i$；

③ 参与各方随机选择ρ_i，其中 $\rho_1 + \rho_2 \cdots + \rho_n = \rho$，通过MPC协议计算；

- ✓ $\rho \cdot (m + rx)$
- ✓ $k \cdot \rho$

在各方分别持有a_i，b_i的情况下，计算出$a \cdot b$

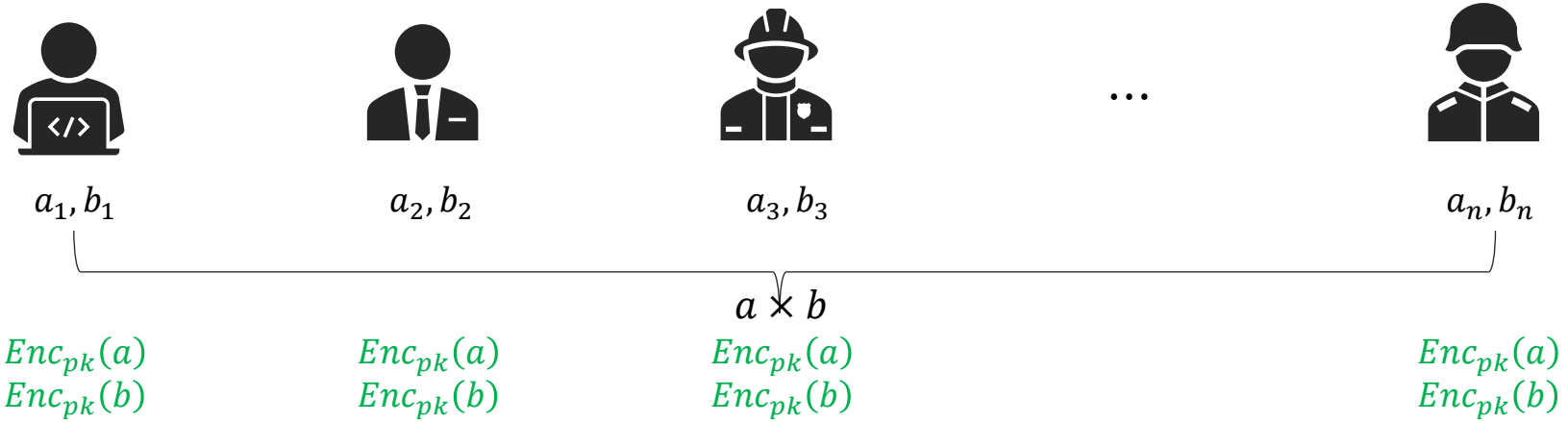
④ 计算出$(k \cdot \rho)^{-1} \cdot \rho \cdot (m + rx) \ mod \ q = s$

门限ECDSA签名

分布式签名： Distributed Signing

$$a = a_1 + a_2 + \cdots + a_n$$
$$b = b_1 + b_2 + \cdots + b_n$$

$a \times b$



各方分别持有a_i、b_i，共同计算$a \times b$；

① 各方分别计算出各自持有值的密文$Enc_{pk}(a_i)$、 $Enc_{pk}(b_i)$，并将密文发送给其他所有方。这里$Enc(\cdot)$为同态加密运算；

② 各自计算出$Enc_{pk}(a) = Enc_{pk}(a_1) + Enc_{pk}(a_2) + \cdots + Enc_{pk}(a_n)$，以及$Enc_{pk}(b) = Enc_{pk}(b_1) + Enc_{pk}(b_2) + \cdots + Enc_{pk}(b_n)$，（同态加法性质）；

③ 各方分别计算$Enc_{pk}(a_i \cdot b) = a_i \cdot Enc_{pk}(b)$，并将所计算的$Enc_{pk}(a_i \cdot b)$发送给其他所有方；

④ 各方计算出$C = Enc_{pk}(a \cdot b) = Enc_{pk}(a_1 \cdot b) + Enc_{pk}(a_2 \cdot b) + \cdots + Enc_{pk}(a_n \cdot b)$；

⑤ 解密获得$c = a \cdot b$。

通常加上ZKP确保各方发送的密文正确

各自使用私钥碎片x_i解密出C的一部分，然后相加：

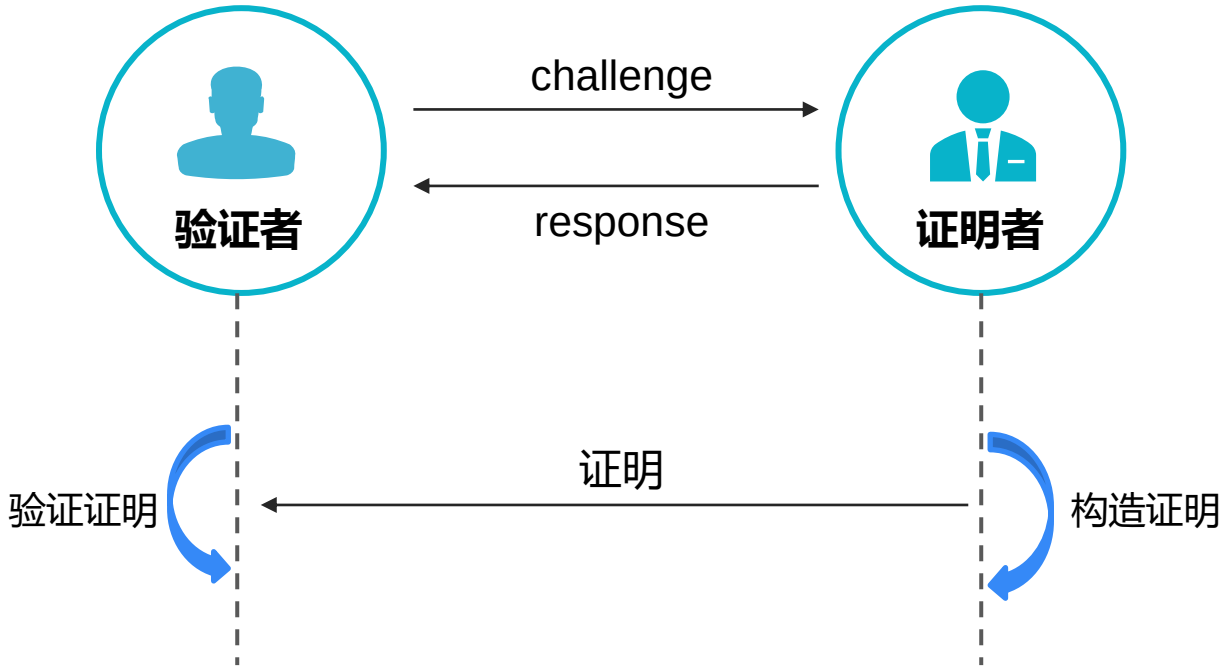
$$c = Dec_{sk_1}(C) + Dec_{sk_2}(C) + \cdots + Dec_{sk_n}(C)$$

3 零知识证明

零知识证明

零知识证明（Zero-Knowledge Proof，ZKP）是密码学中的其中一重要研究技术，它可在不透露原始数据信息的前提下，仍能够使得其他方验证此信息的有效性。

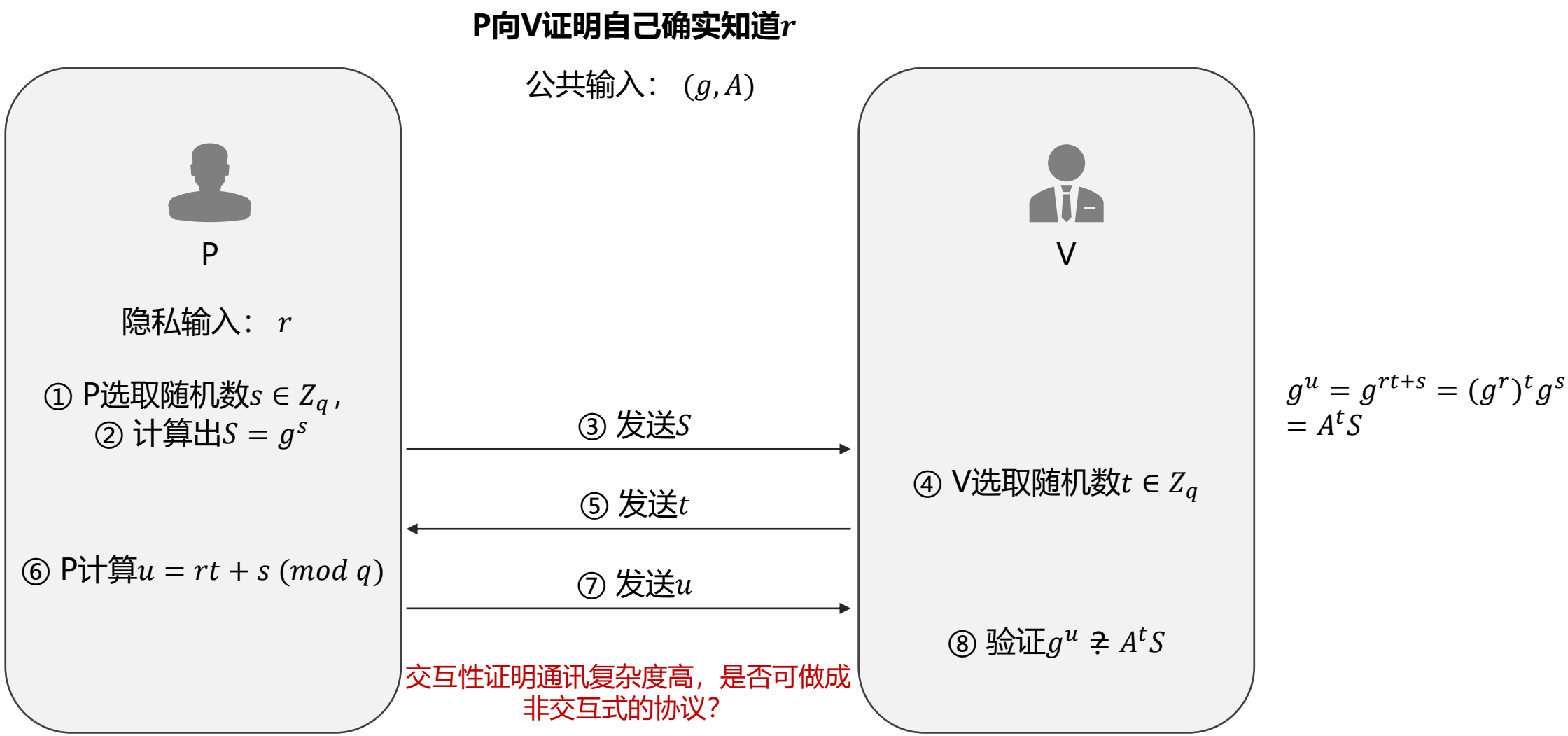
总的来说，零知识证明方案涉及两个对象：证明者和验证者。证明者所起作用为，生成和隐私数据相关的一个声明的证明，并将生成的证明发送给验证者进行验证。验证者除了能知道声明本身的正确性外，无法知道任何其它有效信息。这使得零知识证明在许多实际应用场景下均有很好的应用前景，比如金融、供应链以及身份管理等。



> *我不能告诉你我的秘密，但我可以向你证明我知道这个秘密。*

零知识证明：Schnorr协议

Schnorr零知识协议：对于离散对数问题$A = g^r \ mod \ q$，给定(g, q, A)，证明者能够在不透露r的情况下，向验证者证明自己确实知道r。

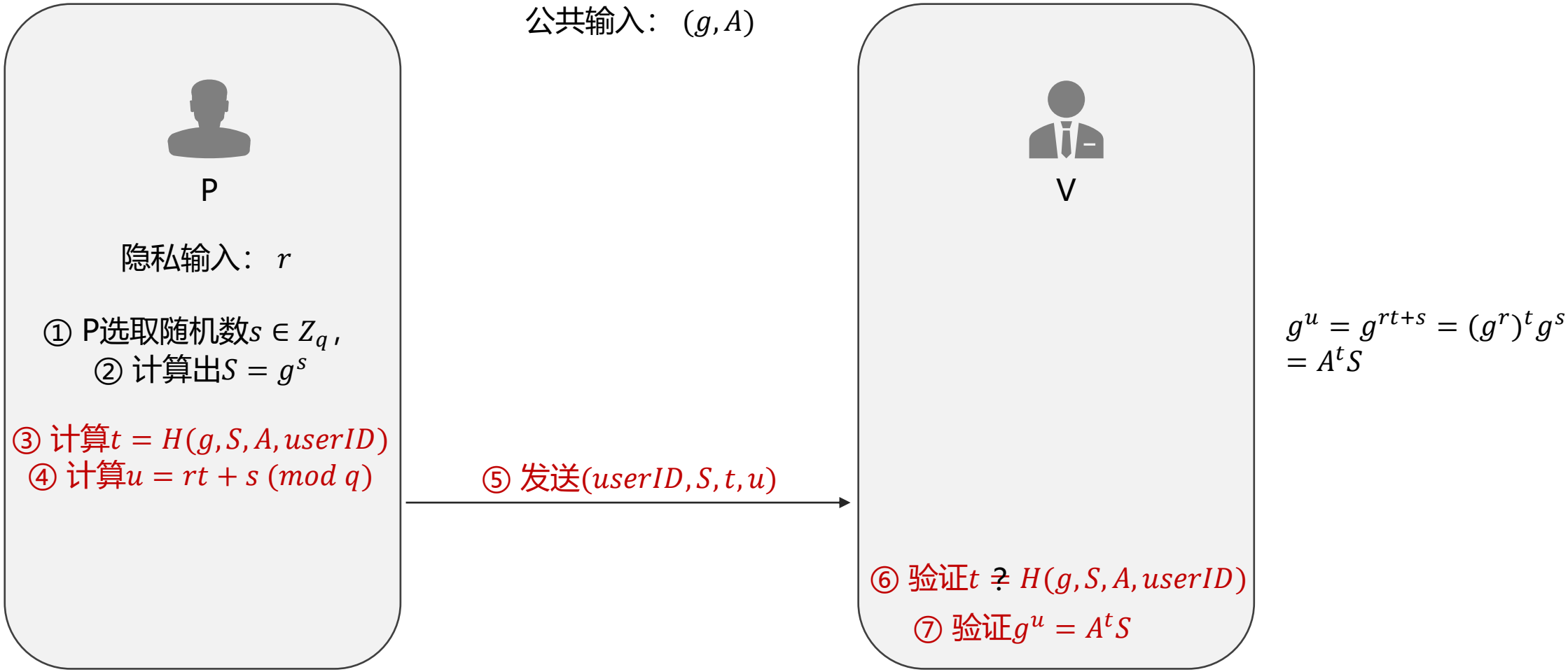


零知识证明：Schnorr NIZK协议

Schnorr零知识协议：对于离散对数问题$A = g^r \ mod \ q$，给定(g, q, A)，证明者能够在不透露r的情况下，向验证者证明自己确实知道r。

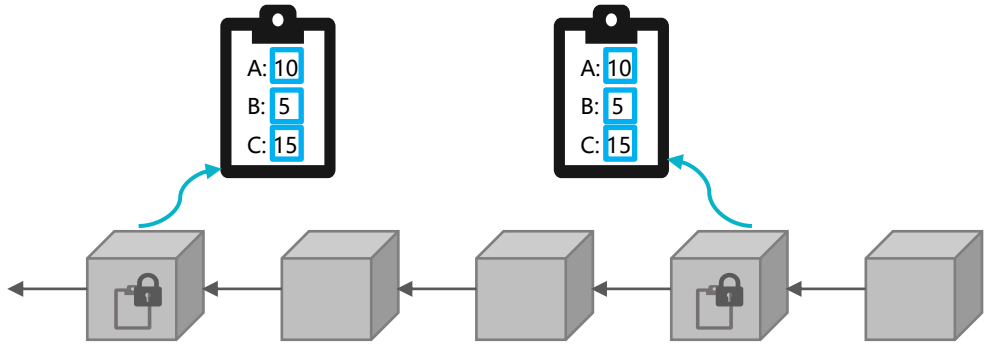
P向V证明自己确实知道r

公共输入：(g, A)



隐私交易：同态加密+零知识证明

- 用加法同态加密余额
- 创建一个交易：用 pk_A和 pk_B加密转账金额，并生成一个零知识证明 π
- 零知识证明如下事实：“两个密文拥有相同的明文，并且该明文不小于0，不大于V_A
- 其他节点同态更新账户A和账户B的余额



A向B发起转账金额为m的交易

Alice (pk_A, sk_A)

(pk_B, sk_B) Bob

余额V_A

余额V_B

1. $m_A \leftarrow Enc(pk_A, m), m_B \leftarrow Enc(pk_B, m)$

2. $\pi \leftarrow ZKP(sk_A, m_A, m_B, V_A)$

3. 发送m_A, m_B, π

4. 通过π验证交易是否确实有效

5. 同态操作更新账户余额。

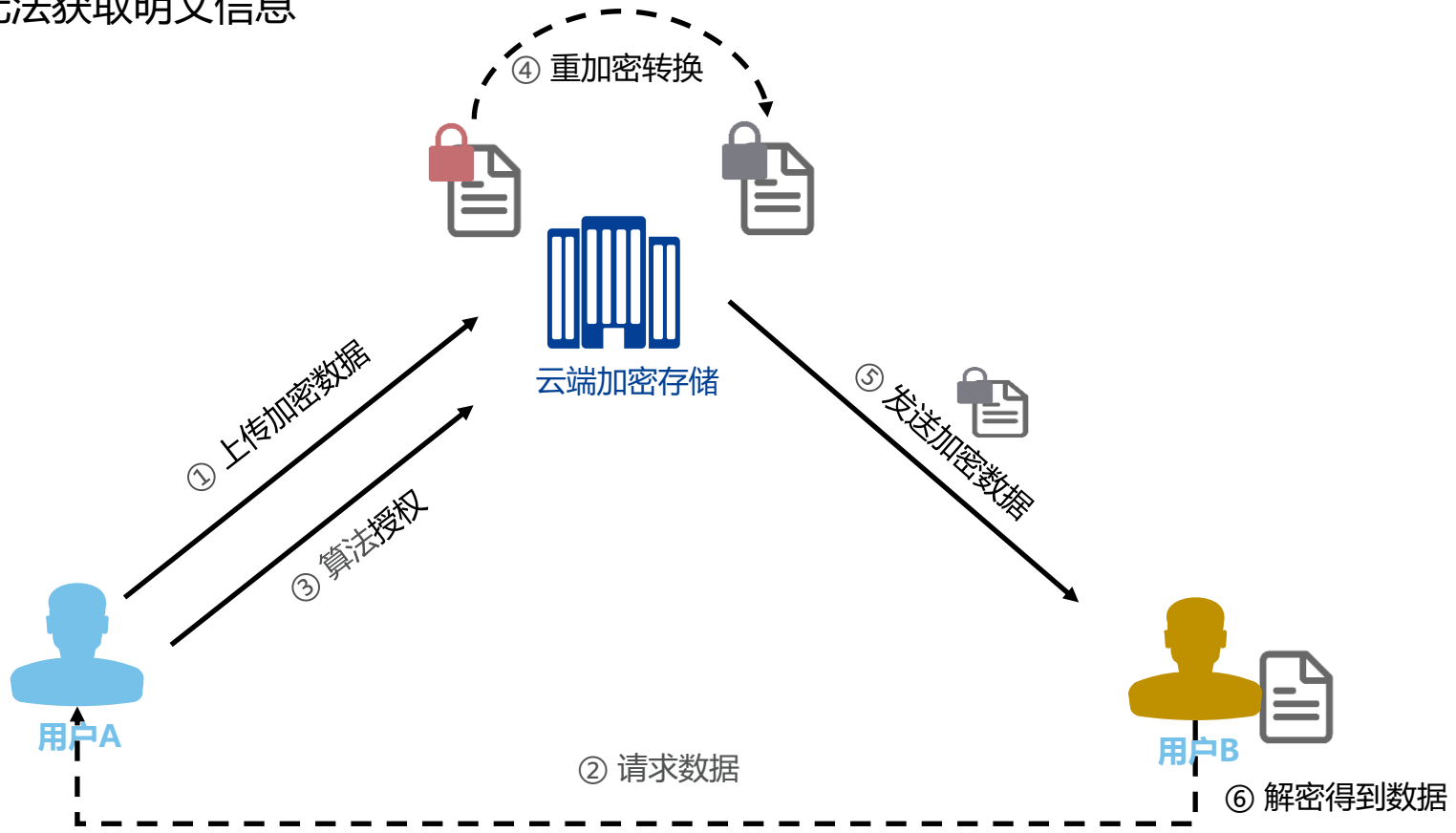
$Eval(pk_A, m_A, Enc(pk_A, V_A)$

$Eval(pk_B, m_B, Enc(pk_B, V_B)$

4 代理重加密

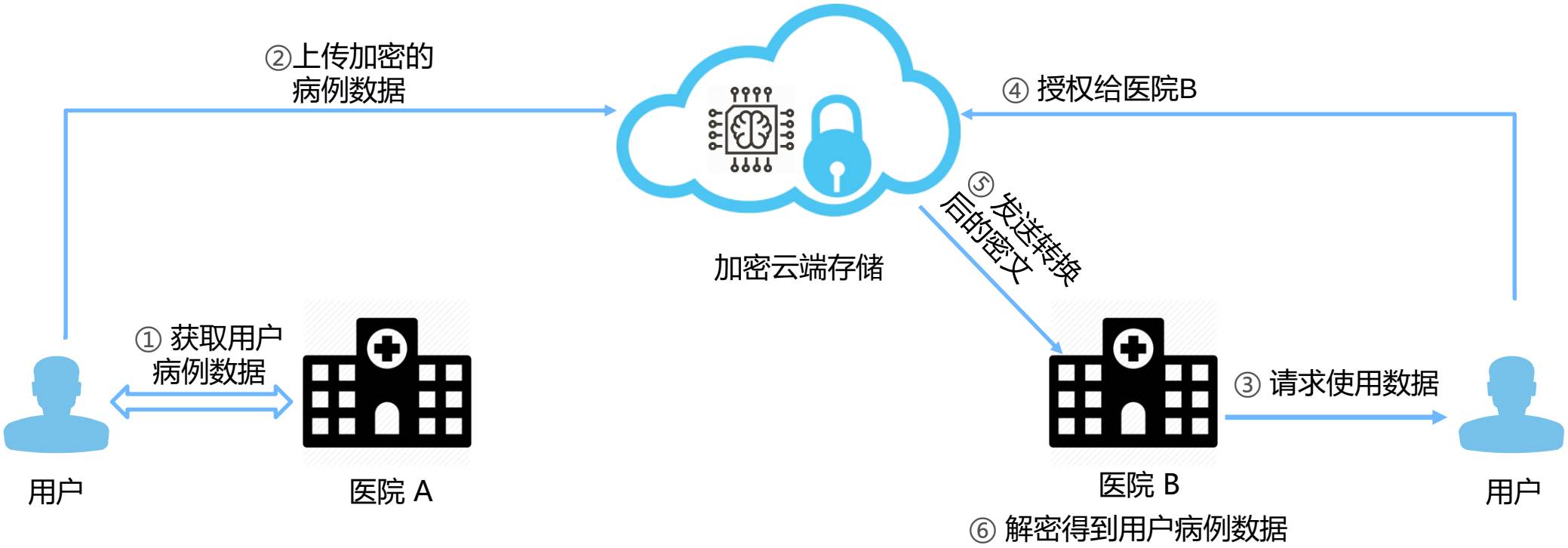
代理重加密

- 数据从A的密文转换为B的密文（重加密过程）
- 云端只做密文转换，无法获取明文信息

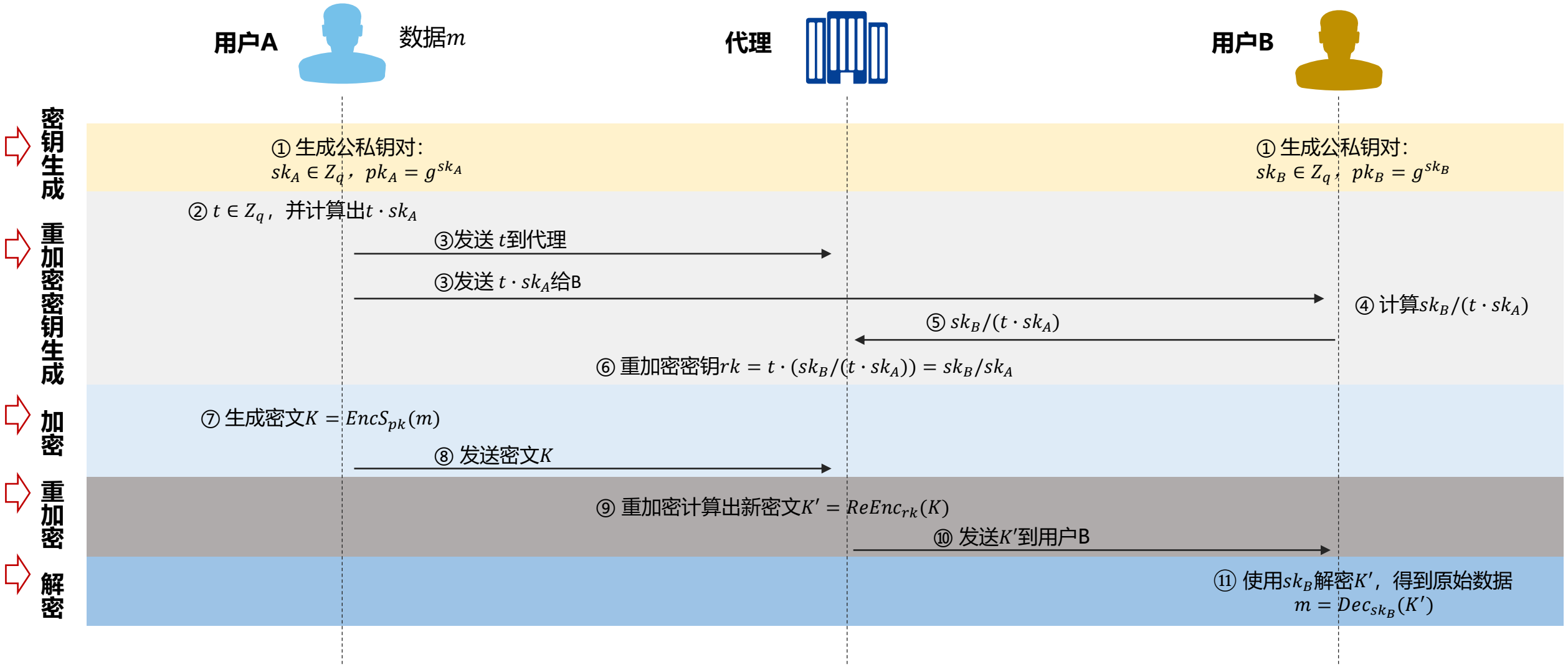


数据授权共享

- 云端提供加密存储服务，可以让病例数据在各医院间共享，云端无法直接获取数据。
- 数据授权完全由用户自己控制，并按照其意愿在不同医院之间共享。
- 用户无需随身携带数据，数据的共享既安全又便利。



代理重加密：算法原理


用户A
数据m
代理
用户B
密钥生成
重加密密钥生成
加密
重加密
解密
① 生成公私钥对：
$sk_A \in Z_q$， $pk_A = g^{sk_A}$
① 生成公私钥对：
$sk_B \in Z_q$， $pk_B = g^{sk_B}$
② $t \in Z_q$，并计算出$t \cdot sk_A$
③发送 t到代理
③发送 $t \cdot sk_A$给B
④ 计算$sk_B/(t \cdot sk_A)$
⑤ $sk_B/(t \cdot sk_A)$
⑥ 重加密密钥$rk = t \cdot (sk_B/(t \cdot sk_A)) = sk_B/sk_A$
⑦ 生成密文$K = EncS_{pk}(m)$
⑧ 发送密文K
⑨ 重加密计算出新密文$K' = ReEnc_{rk}(K)$
⑩ 发送K'到用户B
⑪ 使用sk_B解密K'，得到原始数据
$m = Dec_{sk_B}(K')$

隐私计算技术对比

	安全多方计算 (MPC)	同态加密 (HE)	代理重加密 (PRE)	可信执行环境 (TEE)
概述	• 多方共同参与，在保证输入数据隐私的前提下，完成计算， • 计算结果可按需分配	• 任何人可在密文下进行计算，得到的结果仍然为密文 • 只有私钥所有者可以解密	• 在授权允许下，可将A的密文高效转换为B的密文 • 转换者无法获取明文的任何信息	• 在相对封闭和隔离的环境下进行计算 • 依赖于硬件的安全性，以及对于硬件厂商的信赖
通讯	复杂度高	复杂度低	复杂度低	复杂度低
计算	复杂度低	复杂度高	复杂度低	复杂度较低
安全	算法可证明安全	算法可证明安全	算法可证明安全	取决于硬件系统实现 目前出现过较为严重的安全隐患
背书	算法	算法	算法	厂商信用
可扩展性	取决于带宽	取决于计算能力	可扩展性强	安全环境受限于物理内存环境 很难扩展
应用范围	适用于所有计算	适用于所有计算	仅适用于存储与共享	适用于所有运算

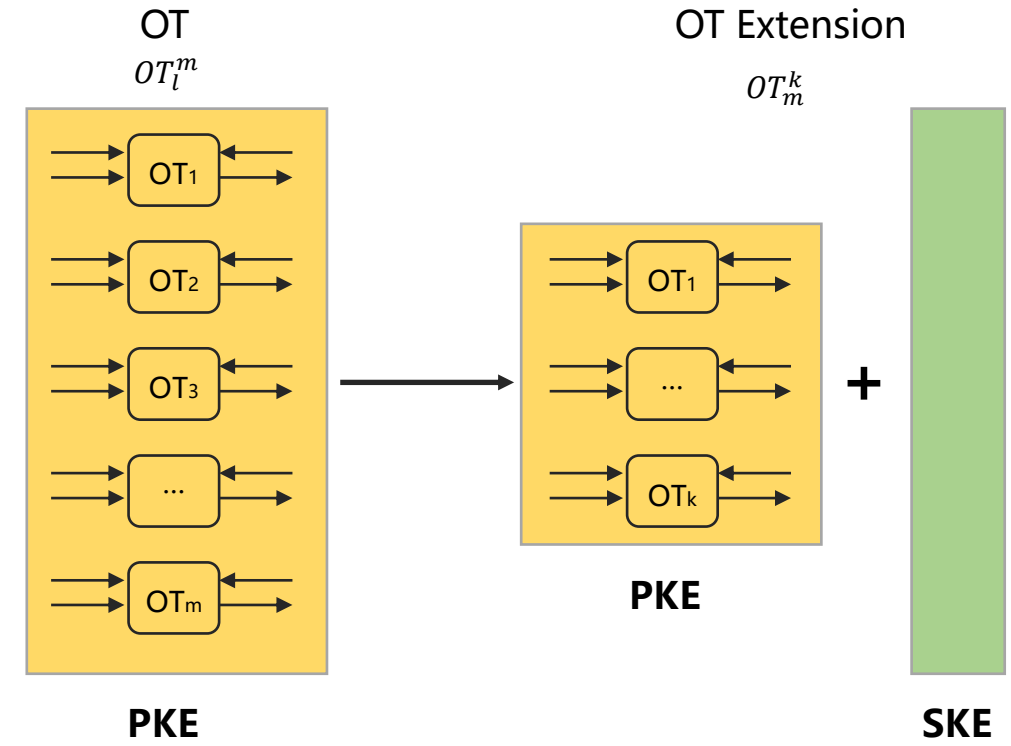
一切皆可计算！

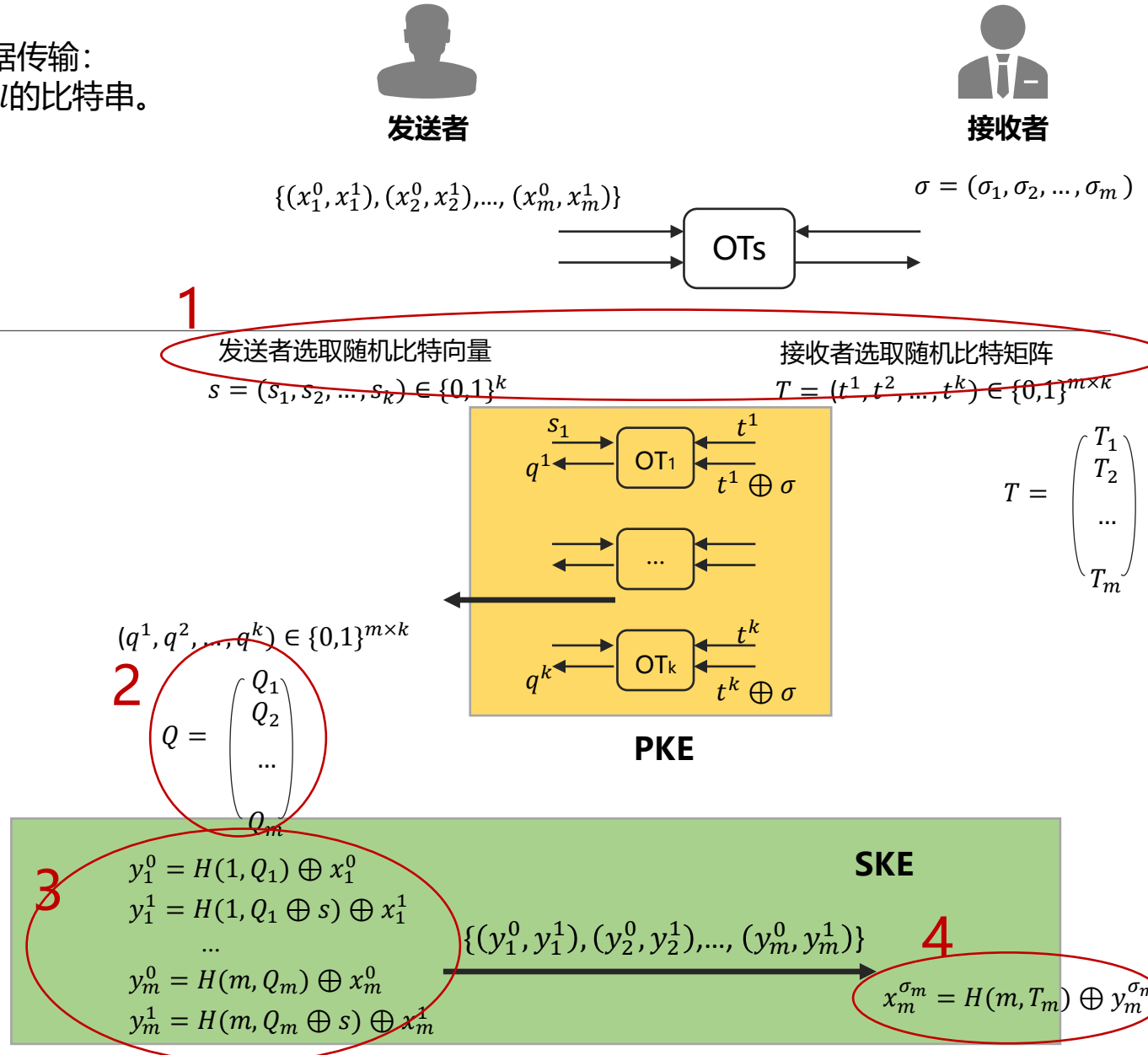
Backup Slides

相关原理：IKNP-OT Extension

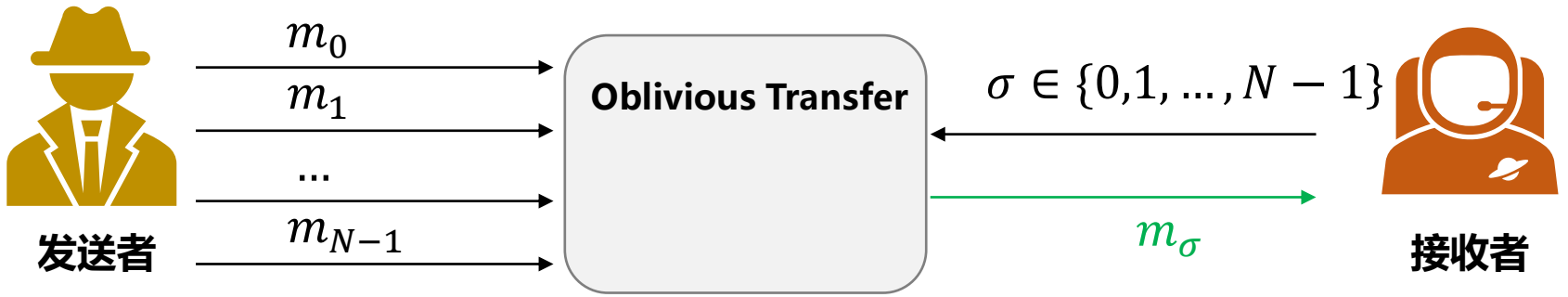
通用不经意传输（OT）定义：OT_l^m可实现m个长度为l比特串的数据传输：
输入：发送者持有m个数据对$(x_i^0, x_i^1), 1 \le i \le m$，这里$x_i^b$是长度为$l$的比特串。
接收者持有m个比特值$\sigma = (\sigma_1, \sigma_2, \dots, \sigma_m)$
输出：接收者得到输出结果$x_i^{r_i}, 1 \le i \le m$，而发送者无任何输出。

公共输入：安全参数k
预言机（哈希函数）$H: [m] \times \{0,1\}^k \to \{0,1\}^l$





相关原理：1-out-of-N OT



① 选取$C_1, C_2, \dots, C_{N-1} \in Z_q$

② 选取$r \in Z_q$，计算g^r，以及C_i^r，$1 \le i \le N-1$

③ 发送g^r，$C_1, C_2, \dots, C_{N-1}$ → ④ 生成公私钥对（$sk_\sigma = k \in Z_q, pk_\sigma = g^k$）

⑤ 发送pk_σ

⑥ 计算pk_σ^r，以及$pk_i^r = C_i^r / pk_\sigma^r$,其中$1 \le i \le N-1$

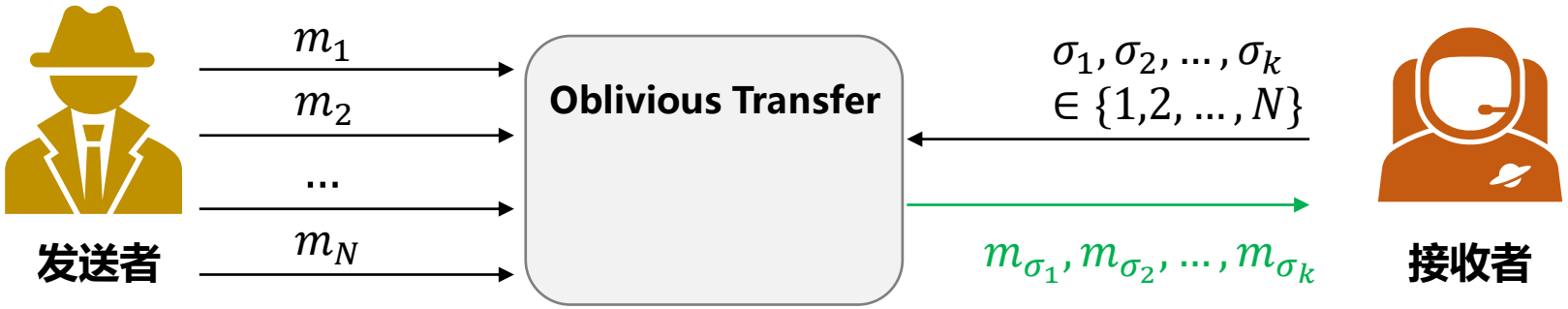
⑦ 选择随机串R，计算出对于m_i的密文$H(pk_i^r, R, i) \oplus m_i$，其中$0 \le i \le N-1$

⑧ 发送R和$H(pk_i^r, R, i) \oplus m_i$ → ⑨ 计算$pk_\sigma^r = (g^k)^r = (g^r)^k = (g^r)^{sk_\sigma}$

$H(pk_\sigma^r, R, \sigma) \longrightarrow m_\sigma$

[1] Naor-Pinkas (2000). *Efficient oblivious transfer protocols.*

相关原理：k-out-of-N OT



两个随机预言机:

$H_1: \{0,1\}^* \rightarrow G_q$

$H_2: G_q \rightarrow \{0,1\}^l$

① 计算$w_{\sigma_j} = H_1(\sigma_j)$，$1 \le j \le k$

$A_j = w_{\sigma_j} g^{a_j}$，$a_j \in Z_q^*$

② 发送A_j，$1 \le j \le k$

③ 选取$x \in Z_q^*$，计算$y = g^x$，$D_j = A_j^x$

④ 计算$w_i = H_1(i)$，$1 \le i \le N$

$c_i = m_i \oplus H_2(w_i^x)$

⑤ 发送y，D_j，c_i

⑥ 计算$k_j = \frac{D_j}{y^{a_j}} = \frac{D_j}{(g^x)^{a_j}} = (A_j / g^{a_j})^x = {w_{\sigma_j}}^x$

⑦ 计算$m_{\sigma_j} = c_{\sigma_j} \oplus H_2(k_j)$

[1] Chu-Tzeng (2005). *Efficient k-out-of-n Oblivious Transfer Schemes with Adaptive and Non-Adaptive Queries.*

相关原理：CO-OT

1-out-of-2 Oblivious Transfer

群：$< G, q, g >$；随机预言机： H

m_0

m_1

发送者

$\sigma = 0/1$

接收者

① 随机选取$a \leftarrow Z_q$，计算$A = g^a$

② 发送A

③ 随机选取$b \leftarrow Z_q$

④ 计算$B = A^\sigma g^b$

$B^a = (A^\sigma g^b)^a$

$B = g^b, if\ \sigma = 0$

$B = Ag^b, if\ \sigma = 1$

⑤ 发送B

⑥ 计算密钥：$k_0 = H(B^a)$， $k_1 = H((\frac{B}{A})^a)$

⑦ 计算密文： $c_0 = k_0 \oplus m_0$， $c_1 = k_1 \oplus m_1$

⑧ 发送c_0， c_1

⑨ 计算：$k_\sigma = H\left(\left(\frac{B}{A^\sigma}\right)^a\right) = H(A^b)$

⑩ 获得：$m_\sigma = k_\sigma \oplus c_\sigma$

[1] Chou-Orlandi (2015). *The Simplest Protocol for Oblivious Transfer.*

相关原理：CO-OT

1-out-of-n Oblivious Transfer

m组$N-1$个l-bits

$\{M_0^i, \dots, M_{N-1}^i\}_{i\in[m]}$

发送者

椭圆曲线上的群：$< G, B, p, + >$；随机预言机：
$H: (G \times G) \times G \to \{0,1\}^k$

$c^1, \dots, c^m$

接收者

① 随机选取$y \leftarrow Z_p$，计算：$S = yB$，$T = yS$

② 发送S

③ 验证$S \in G$

④ 随机选取$x^i \leftarrow Z_p$，$i \in [m]$

⑤ 计算$R^i = c^i S + x^i B$，$i \in [m]$

$yR^i = (c^i S + x^i B)y$

⑥ 发送R^i，$i \in [m]$

⑦ 验证$R^i \in G$，$i \in [m]$

⑧ 对于$j \in [N]$，计算密钥：$k_j^i = H(S, R^i, yR^i - jT)$

⑨ 对于$j \in [N]$，计算密文：$C_j^i = k_j^i \oplus m_j^i$

⑩ 发送C_j^i

⑪ 计算$k_{c^i}^i = H(S, R^i, yR^i - c^i T)$
$= H(S, R^i, x^i By) = H(S, R^i, x^i S)$

⑫ 解密$M_{c^i}^i = k_{c^i}^i \oplus C_{c^i}^i$

[1] Chou-Orlandi (2015). *The Simplest Protocol for Oblivious Transfer.*

相关原理：ALSZ-OT

General 1-out-of-2 Oblivious Transfer

群：$< G, q, g >$；随机预言机： H

n组2个l-bits

$\{x_i^0, x_i^1\}_{i\in[n]}$

发送者

$\binom{2}{1}$-OT_l^n

$\sigma = (\sigma_1, \dots, \sigma_n)$

接收者

① 随机选取$\alpha_i \leftarrow Z_q$， $h_i \leftarrow G$， $i \in [n]$

② 计算$(h_i^0, h_i^1) \stackrel{\text{def}}{=} \begin{cases} (g^{\alpha_i}, h_i) & if\ \sigma_i = 0 \\ (h_i, g^{\alpha_i}) & if\ \sigma_i = 1 \end{cases}$

③ 发送(h_i^0, h_i^1)， $i \in [n]$

④ 随机选取$r \leftarrow Z_q$， 计算$u = g^r$

⑤ 计算密钥：$k_i^0 = (h_i^0)^r$， $k_i^1 = (h_i^1)^r$

⑥ 计算密文：

$v_i^0 = x_i^0 \oplus KDF(k_i^0)$， $v_i^1 = x_i^1 \oplus KDF(k_i^1)$

⑦ 发送u以及(v_i^0, v_i^1)， $i \in [n]$

⑧ 计算：$k_i^{\sigma_i} = (h_i^{\sigma_i})^r = u^{\alpha_i}$

⑨ 计算：$x_i^{\sigma_i} = v_i^{\sigma_i} \oplus KDF(u^{\alpha_i})$

[1] Asharov-Lindell-Schneider-Zohner (2013). *More Efficient Oblivious Transfer and Extensions for Faster Secure Computation.*

相关原理：ALSZ-OT Extension*

群：$< G, q, g >$；PRG G；随机预言机： H

$\{x_i^0, x_i^1\}_{i\in[n]}$

发送者

$\binom{2}{1}$-OTE_l^n

$\sigma = (\sigma_1, \dots, \sigma_n)$

接收者

② Choose a random bit string $s = (s_1, s_2, \dots, s_k) \in \{0,1\}^k$

s_i

$\binom{2}{1}$-OT_k^k

r_i^0

r_i^1

① choose k pairs k-bits seeds $\{(r_i^0, r_i^1)\}_{i=1}^k$

③ outputs $r_i^{s_i}$

④ 取$T = [t^1|\dots|t^k]$，$t^i = G(r_i^0)$，
T的第i列为t^i $(1 \le i \le k)$； T的第j行为t_j $(1 \le j \le n)$；

⑤ 计算$u^i = t^i \oplus G(r_i^1) \oplus \sigma$

⑥ 发送u^i， $1 \le i \le k$

⑦ 计算$q^i = (s_i \cdot u^i) \oplus G(r_i^{s_i})$， $1 \le i \le k$，
令$Q = [q^1|\dots|q^k]$，Q的第i列为q^i $(1 \le i \le k)$；
Q的第j行为q_j $(1 \le j \le n)$；

⑧ 计算$y_j^0 = x_j^0 \oplus H(j, q_j)$， $y_j^1 = x_j^1 \oplus H(j, q_j \oplus s)$

⑨ 发送y_j^0, y_j^1， $1 \le j \le n$

⑩ 计算$x_j^{r_j} = y_j^{r_j} \oplus H(j, t_j)$， $1 \le j \le n$

[1] Asharov-Lindell-Schneider-Zohner (2013). *More Efficient Oblivious Transfer and Extensions for Faster Secure Computation.*

相关原理：KOS-OT

群：$< G, q, g >$；PRG G；随机预言机： H

$\{x_i^0, x_i^1\}_{i\in[n]}$

发送者

$\binom{2}{1}$-OTE_l^n

$\sigma = (\sigma_1, \dots, \sigma_n)$

接收者

② Choose a random bit string
$s = (s_1, s_2, \dots, s_k) \in \{0,1\}^k$

s_i

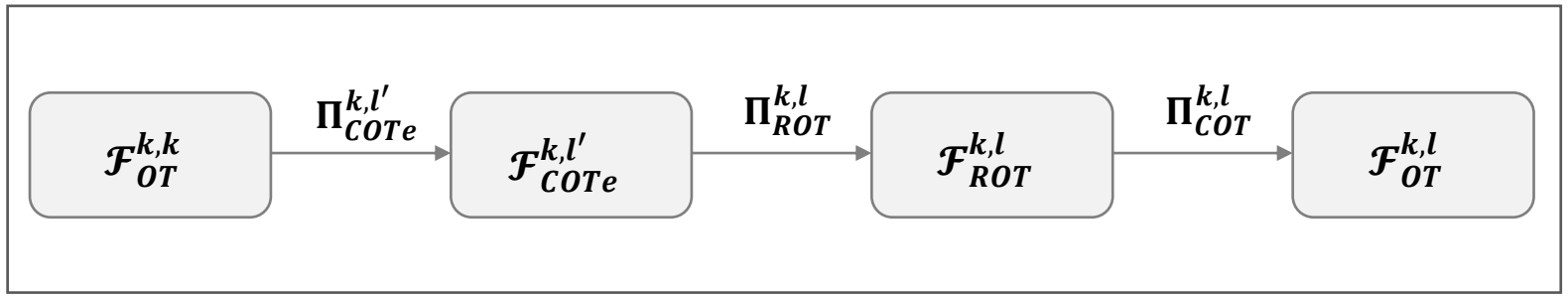
$\binom{2}{1}$-OT_k^k

r_i^0

r_i^1

① choose k pairs k-bits seeds
$\{(r_i^0, r_i^1)\}_{i=1}^k$

③ outputs $r_i^{s_i}$



[1] Marcel Keller , Emmanuela Orsini , Peter Scholl (2015). *Actively Secure OT Extension with Optimal Overhead.*

PSI方案构造

基于OT的PSI方案

$\binom{N}{1}$-OT　　**比较两个长度为σ比特的元素x是否等于y**


P1


P2

输入： $X = \{x_1, x_2, \dots, x_n\}$　　$Y = \{y_1, y_2, \dots, y_m\}$

比较$x_i \triangleq y_j$

判断：$x_i \in Y = \{y_1, y_2, \dots, y_m\}$

确定：$X \cap Y$

$\boldsymbol{x = x[\sigma-1] \dots x[1]x[0]}$　　$\boldsymbol{y = y[\sigma-1] \dots y[1]y[0]}$

① 按比特切分为t块　　① 按比特切分为t块

$\boldsymbol{x = x[t-1]|| \dots ||x[1]||x[0]}$　　$\boldsymbol{y = y[t-1]|| \dots ||y[1]||y[0]}$

$\boldsymbol{N = 2^{\sigma/t}}$

② 选取t组元素个数为N，元素长度为l比特的随机串$(s_0^i, s_1^i, \dots, s_{N-1}^i)$，$1 \le i \le t$

$\boldsymbol{x[i] \in [0, N-1]}$　　$\binom{N}{1}$-OT_l^t

$s_{x[i]}^i$

③ 输出$s_{x[i]}^i$，$1 \le i \le t$

④ 计算$m_1 = \oplus\, s_{x[i]}^i$，$1 \le i \le t$　　④ 计算$m_2 = \oplus\, s_{y[i]}^i$，$1 \le i \le t$

⑤ 发送m_2

⑥ 比较$m_1 \stackrel{?}{=} m_2$

[1] Pinkas-Schneider (2014). *Faster Private Set Intersection based on OT extension.*

MPC工程实现框架

具体到工程实现层面，当前值得我们进一步参考与借鉴的成果。整体上，目前安全多方计算领域的理论研究与工程实现处于比较健康状态，其中两方安全计算的工程实现相对更加成熟。

				安全模型			电路模型		支持数据类型							支持运算类型									
框架名称	协议类型	支持参与方	开发语言	半诚实敌手模型	恶意敌手模型	混合敌手模型	布尔电路	算术电路	布尔	整型	任意整型	浮点	数组	结构体	动态数组	布尔	比较	加法	乘法	除法	移位	开源	开发者文档	示例代码	最后更新日期
EMP-toolkit	GC	2	C++	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	2018.09
Obliv-C	GC	2	C, OCaml	✓	✗	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	2018.06
ObliVM	GC	2	Java	✓	✗	✓	✓	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	2016.02
TinyGarble	GC	2	C/C++	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	2018.10
Wysteria	GMW	2+	OCaml	✓	✗	✓	✓	✗	✓	✓	✗	✗	✗	✓	✗	✗	✓	✓	✓	✗	✗	✓	✓	✓	2014.10
ABY	GC, GMW	2	C++	✓	✗	✓	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓	✗	✗	✓	✓	✓	2018.10
SCALE-MAMA	GC, SS	2+	C++, Python	✓	✓	✓	✗	✓	✗	✓	✓	✓	✓	✓	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	2018.10
Sharemind	SS	3	C/C++	✓	✗	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	2018.09
PICCO	SS	3+	C/C++	✓	✗	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	2017.10
Frigate	BMR	2+	C++	-	-	✗	✓	✗	✗	✓	✓	✗	✓	✓	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	2016.05
CBMC-GC	-	2+	C++	-	-	✗	✓	✗	✓	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	2017.05

- EMP-tookit：半诚实敌手下的通用计算框架
- Obliv-C：较健壮的GC框架，适用于想实现和优化ORAM的开发者
- ABY：适合有密码学背景的开发者
- SCALE-MAMA：支持多方，强安全性保证
- Sharemind：适合企业，支持复杂函数运算
- PICCO：适合对多方计算有需求的场景

代理重加密：构造方案

定义1：一个阶为q的群G，g、h均为其生成元，即$< g >=< h >= G$；双线性映射$e: G \times G \to G_T$；
定义2：两个随机预言机$H: \{0,1\}^{\leq l} \to G_T$和$F: \{0,1\}^{\leq l} \to G_T$，这里$l$为一次签名算法中生成密钥的长度；
定义3：一个强不可伪造的一次签名方案为(G_0, S_0, V_0)，分别为密钥生成，签名生成以及签名验证；
定义4：算法$Check$是对密文数据(A, B, C, D, E, S)以相应的公钥pk，验证密文是否有效，包含以下几个步骤：

- 执行$V_0(A, (C, D, E), S)$，以对应的签名验证A为输入，验证关于消息(C, D, E)的签名S是否有效；
- 校验$e(B, F(A)) = e(pk, D)$以及$e(B, h) = e(pk, E)$；
- 如果全部通过则输出1，否则输出0。

基于双线性映射和随机预言机（哈希函数），一个完整的PRE方案构造如下：

- 1. 密钥生成：对于安全参数k，随机选取$x \in Z_q$，$y = g^x$。x即为生成的私钥sk，公钥pk则为y，即：

$$(sk, pk) = (x, g^x) \leftarrow KeyGen(1^k)$$

则A、B的公私钥对分别为(pk_A, sk_A)、(pk_B, sk_B)；

- 2. 重加密密钥生成：以A的私钥sk_A和B的私钥sk_B为输入，生成重加密密钥：

$$rk = \frac{sk_B}{sk_A} \ mod \ q \leftarrow ReKeyGen(sk_A, sk_B)$$

具体过程是先由A选取随机数$t \in Z_q$，并计算出$t \cdot sk_A$发送给B，同时将t发送给代理。B计算出$\frac{sk_B}{t \cdot sk_A}$发送给代理，则代理计算出$rk = \frac{sk_B}{sk_A} \ mod \ q$。

- 3. 加密：以消息$M \in G_T$和用户A的公钥pk_A为输入：
  - 通过一次签名方案的密钥生成算法生成一对签名密钥对(ssk, svk)，ssk为签名密钥，svk为签名验证密钥，并令$A = svk$；

$$(ssk, svk) \leftarrow G_0(1^k)$$

  - 选取随机数$r \in Z_q$，计算出：
    $B = (pk_A)^r$，$C = e(g, H(A))^r \cdot m$，$D = F(A)^r$，$E = h^r$
  - 执行签名算法：$S \leftarrow S_0(ssk, (C, D, E))$
  - 最终获得的数据密文为$K = (A, B, C, D, E, S)$
- 4. 重加密：以重加密密钥rk为输入，执行重加密过程，将公钥pk_A下的密文(A, B, C, D, E, S)转换为公钥pk_B下的密文：
  - 计算$B' = B^{rk} = (pk_A)^{r \cdot rk} = g^{sk_A \cdot r \cdot rk} = g^{sk_A \cdot r \cdot rk} = g^{r \cdot sk_B}$
  - 执行$Check(K, pk_A)$，验证密文是否有效。验证通过则新生成密文为$K' = (A, B', C, D, E, S)$；
- 5. 解密：首先执行$Check(K', pk_B)$，验证通过。则使用sk_B解密K'，得到解密数据$C / e(B', H(A))^{1/sk_B}$，即为m。

Paillier同态加密

算法构成

- **密钥生成：**生成两个等长的大素数p和q，令$N = pq$，计算出$\lambda = lcm(p-1, q-1)$。$g \in Z_{N^2}^*$是$Z_{N^2}^*$中的一个生成元。公钥则为(N, g)，私钥为λ；
- **加密：**对消息$m \in Z_N$加密，选择随机数$r \in Z_N^*$，生成密文$c = g^m \cdot r^N \ mod \ N^2$；
- **解密：**密文$c \in Z_{N^2}^*$，解密获得消息明文$m = \frac{L(c^\lambda \ mod \ N^2)}{L(g^\lambda \ mod \ N^2)} \ mod \ N$，其中$L(u) = \frac{u-1}{N}$。

加法同态性质

给定分别对消息m_1和m_2的密文c_1、c_2，定义同态加法函数$+_E$，$c_1 +_E c_2 = c_1 c_2 \ mod \ N^2$，$c_i = E_{pk}(m_i)$，则

$$c_1 +_E c_2 = E_{pk}(m_1 + m_2);$$

并且，对于$k \in Z_N$，$k \times_E c = c^k \ mod \ N^2$，则

$$k \times_E c = E_{pk}(m)^k$$

zk-SNARK