> 本文主要记录隐私计算中的不经意传输（Oblivious Transfer，OT）相关技术，包括早期的 **Rabin-OT** 以及常见的 **1-out-of-2 OT** 和 **chosen 1-out-of-2 OT** 等各类技术，**仅供参考**。

一、OT分类

首先介绍一下常见的几种OT技术的特点。

1. Rabin-OT

1981年 Rabin [1] 首次提出OT的概念，并构造了一个OT协议（Rabin-OT），该协议中存在一个发送方和一个接收方，发送方拥有一个消息 m，协议保证接收方能**以 1/2 的概率**获得消息 m，且发送方**不了解**接收方是否获得消息。

- **发送方的输入**：消息 m
  接收方的输入：无
- **发送方的输出**：无
  接收方的输出：1/2 的概率获得消息 m
- **信息**：接收方知道自己是否获得消息，而发送方不知情。

2. 2选1OT（1-out-of-2 OT）

1985年 Even、Goldreich 和 Lempel [2] 三人提出2选1OT的概念，该协议中存在一个发送方和一个接收方，发送方拥有两个消息 m_0, m_1，协议保证接收方能**以同等的概率**（1/2）获得其中的一个消息，且发送方**不了解**接收方获得了哪一个消息。

- **发送方的输入**：消息 m_0, m_1
  接收方的输入：无
- **发送方的输出**：无
  接收方的输出：获得消息 m_0 或 m_1 （概率同等）
- **信息**：接收方无权选择，但知道自己获得了哪个消息，对另外一条消息的内容不了解。发送方不知道接收方收到了哪条消息。

3. 选择2选1OT（chosen 1-out-of-2 OT）

事实上，仅需对Even三人提出的2选1OT进行一些改变就可以得到选择2选1OT协议，该协议与2选1OT的区别是接收方可以主动选择接收哪一条消息。

- **发送方的输入**：消息 m_0, m_1
  接收方的输入：$c \in \{0, 1\}$ 代表要接收的消息索引
- **发送方的输出**：无
  接收方的输出：获得消息 m_c
- **信息**：接收方可以选择获得哪个消息，但对另外一条消息的内容不了解。发送方不知道接收方收到了哪条消息。

4. 随机OT（Random OT）

1995年 Beaver [3] 通过预处理优化OT的效率，在预处理阶段间接引入了随机OT的概念，该协议不接收输入，输出给发送方两个随机消息 m_0, m_1，给接收方其中一个消息 m_r。

- **发送方的输入**：无

接收方的输入：无

- **发送方的输出**：随机消息 m_0, m_1
  接收方的输出： $r \in \{0,1\}$ 和 m_r
- **信息**：接收方知道自己获得了哪个消息，对另外一条消息的内容不了解。发送方不知道接收方收到了哪条消息。

5. n选1OT 与 OT扩展技术

对于n选1OT而言，发送方的消息数量变成了 n 个，而其余的部分与2选1OT类似。事实上，如今我们在隐私计算或密码学中提到的OT通常是指**可选择的OT**，一般不会再对所谓的 “1-out-of-2 OT” 以及 “chosen 1-out-of-2 OT” 进行区分。因此对于n选1OT而言，一般是指 chosen 1-out-of-n OT。

在实际的安全多方计算协议中可能会调用非常多次的OT协议，逐个单独调用非常影响计算效率，因此就出现了OT扩展技术，能够实现OT并行执行的效果。本文主要介绍一些基础的OT技术，OT扩展技术暂时不作介绍。

二、Rabin-OT 协议

Rabin-OT 协议主要基于**二次剩余的平方根计算**问题构造而来，在给出具体协议流程前先介绍一下二次剩余。

1. 二次剩余

> **定义**：对于整数 a, n，若存在整数 x，满足 $x^2 \equiv a \ (mod\ n)$，则称 a 为模 n 的**二次剩余**，否则称 a 为模 n 的**二次非剩余**。

求解二次剩余的平方根有许多算法，这里不作记录，主要介绍一下 Rabin-OT 会用到的二次剩余的一些常见特性：

特性1：若 n 为奇素数且 $a \not\equiv 0 \ (mod\ n)$，则显然 x 有**两个解** $\pm x_0$（模 n 意义下）。

特性2：若 $n = pq$（其中 p, q 为两个大素数），且 $gcd(a, n) = 1$，则方程 $x^2 \equiv a \ (mod\ n)$ 存在**四个解**。

分析：显然由方程 $x^2 \equiv a \ (mod\ n)$ 可以得到两个同余方程：

$$\begin{cases} x^2 \equiv a \ (mod\ p) \\ x^2 \equiv a \ (mod\ q) \end{cases}$$

由特性1可知，每个方程可以得到两个解，记为 $\pm x_p, \pm x_q$。由**中国剩余定理**（之前文章中介绍过）可知，两个同余方程的每一对解可以**唯一确定**在模 n 意义下的一个解，因此可以得到四个解 x_0, x_1, x_2, x_3，即：

$$\begin{cases} x_0 = CRT(x_p, x_q) \\ x_1 = CRT(x_p, -x_q) \\ x_2 = CRT(-x_p, x_q) \\ x_3 = CRT(-x_p, -x_q) \end{cases}$$

特性3：在特性2的背景下，若 x_0 为方程 $x^2 \equiv a \ (mod\ n)$ 的一个解，则所有解为 $\pm x_0, \pm r x_0$（其中 $\pm r$ 为模 n 的**非平凡二次单位根**）。

分析：模 n 的二次**单位根**指方程 $x^2 \equiv 1 \ (mod\ n)$ 的四个解（由特性2可知一定有四个解），且其中两个解为 ± 1，两个解设为 $\pm r$。不为 ± 1 的解称为**非平凡二次单位根**。

例如：$n = 3 \times 5 = 15$ 时，$x^2 \equiv 1 \ (mod\ 15)$ 的四个解为 $\pm 1, \pm 4$，则 ± 4 就是模 n 的非平凡二次单位根。

若 x_0 为方程 $x^2 \equiv a \ (mod\ n)$ 的一个解，那么 x_0 乘上模 n 的二次单位根也一定为一个解：

$$\begin{cases} (1 \cdot x_0)^2 \equiv 1 \cdot x_0^2 \equiv a \ (mod\ n) \\ (-1 \cdot x_0)^2 \equiv 1 \cdot x_0^2 \equiv a \ (mod\ n) \\ (r \cdot x_0)^2 \equiv 1 \cdot x_0^2 \equiv a \ (mod\ n) \\ (-r \cdot x_0)^2 \equiv 1 \cdot x_0^2 \equiv a \ (mod\ n) \end{cases}$$

2. 协议流程

Rabin-OT 的具体协议流程如下：

- **输入**：发送方的消息 m
  输出：接收方以 1/2 的概率获得消息 m

1. 发送方选择两个**大素数** p, q 并计算 $n = pq$，将n 发送给接收方。
2. 接收方随机选择 x_0 满足 $gcd(x_0, n) = 1$，并计算 $a = x_0^2 \; mod \; n$，将 a 发送给发送方。
3. 发送方求解方程 $x^2 \equiv a \; (mod \; n)$，并得到四个解 $\pm x_0, \pm rx_0$，其中 $\pm r$ 为模 n 的非平凡二次单位根。发送方**随机挑选**一个解（记为 x_1）发送给接收方。
4. 接收方检查，若 $x_1 \not\equiv \pm x_0 \; (mod \; n)$（**概率为1/2**），则可计算出 $d = gcd(x_0 - x_1, n)$，其中 $d = p$ 或 $d = q$，从而获取到发送方生成的两个大素数 p, q。
5. 发送方以 n 作为公钥加密消息 m，并将密文发送给接收方（这里不关注具体的加密算法，可以是RSA加密，只要保证**拥有** p, q **能解密**即可）。
6. 若接收方在第4步获取到 p, q（概率为1/2），则解密获得消息m。

分析：协议的重点也是难以理解的地方是第4步，为什么只有当 $x_1 \not\equiv \pm x_0 \; (mod \; n)$ 时，计算出的 $d = gcd(x_0 - x_1, n)$ 一定等于 p 或者 q ?

- 显然，当 $x_1 \equiv x_0 \; (mod \; n)$ 时，$d = gcd(x_0 - x_1, n) = n$；
  当 $x_1 \equiv x_0 \; (mod \; n)$ 时，$d = gcd(x_0 - x_1, n) = gcd(2x_0, n) = 1$。

- 当 $x_1 \not\equiv \pm x_0 \; (mod \; n)$ 时，考虑到 $x_1 = \pm r \cdot x_0$ 且 $gcd(x_0, n) = 1$，此处**不妨设** $x_1 = r \cdot x_0$，则可以得到：

$$d = gcd(x_0 - x_1, n) = gcd(x_0 \cdot (1 - r), n) = gcd(1 - r, n)$$

由于 $gcd(1 - r, n) \neq 0$，则 $1 - r$ **一定不为** n **的倍数**，接下来我们只要**证明** $1 - r$ **一定为** p **或** q **的倍数**，就可以得到 $d = gcd(1 - r, n)$ 一定等于 p 或者 q。

- 在求解 $x^2 \equiv 1 \; (mod \; n)$ 时我们会首先求解 $x_p^2 \equiv 1 \; (mod \; p)$ 和 $x_q^2 \equiv 1 \; (mod \; q)$ 的解，再利用中国剩余定理进行组合，显然此处 $x_p = \pm 1$，$x_q = \pm 1$，一共有四种组合方式，对应着四个二次单位根。

- 显然其中两个**平凡**二次单位根，即 ± 1 对应着 x_p, x_q 两个**同符号**的组合：
  当 $x \equiv 1 \; (mod \; n)$ 时，$x_p = 1 \; mod \; p = 1$， $x_q = 1 \; mod \; q = 1$ 同为正号；
  当 $x \equiv -1 \equiv n - 1 \; (mod \; n)$ 时，$x_p = (n-1) \; mod \; p = -1$， $x_q = (n-1) \; mod \; q = -1$ 同为负号；

- 而其中的两个**非平凡**二次单位根，即 $\pm r$ 对应着 x_p, x_q 两个**异符号**的组合，分别为 $(x_p, x_q) = (1, -1)$ 和 $(x_p, x_q) = (-1, 1)$。
  当 $x = r$ 对应 $(x_p, x_q) = (1, -1)$ 时，可以得到 $r \; mod \; p = 1$，即 $1 - r$ 为 p 的倍数。
  当 $x = r$ 对应 $(x_p, x_q) = (-1, 1)$ 时，可以得到 $r \; mod \; q = 1$，即 $1 - r$ 为 q 的倍数。

- 因此 $1 - r$ 一定为 p 或 q 的倍数，得证。

当接收方获得 d 为 p 或 q 后，计算 n/d 即可得到另一个素数的值。

三、2选1OT、选择2选1OT、n选1OT协议

我们假设**消息空间**以及加密算法中的**明文空间**为长度为 k 的 01 串，即 $\{0,1\}^k$；加密算法的**密文空间**为长度为 l 的 01 串，即 $\{0,1\}^l$。我们此处不关注具体使用什么样的加密算法，只把加解密当作**黑盒子**使用，拥有公钥能把明文加密成密文，拥有私钥能把密文解密成明文。

这里声明变量为 01 串只是因为协议中会使用到**逐位的异或操作**，记为 $\oplus$。事实上，协议可以将这些变量的定义域以及异或操作推广至更一般的场景，有兴趣的可以查看原论文[2]，此处用 01 串来表示，便于理解。

1. 2选1OT协议

2选1OT协议中会用到加解密，这里为**简化表示**提前声明**发送方拥有公钥和私钥**，**接收方拥有公钥**。加密记为 $Enc(\cdot)$，解密记为 $Dec(\cdot)$。此外流程中**不对**生成的随机串的**长度进行说明**，要在明文空间运算的长度就为 k，在密文空间运算的长度就为 l。协议具体流程如下：

- **输入**：发送方提供消息 m_0, m_1
  输出：接收方获得消息 m_0 **或** m_1 （概率同等）

1. 发送方生成两个随机串 r_0, r_1 并发送给接收方。
2. 接收方生成随机串 k，以及一个随机比特 $c \in \{0,1\}$，计算 $q = Enc(k) \oplus r_c$，并将 q 给发送方。
3. 发送方计算 $k_0 = Dec(q \oplus r_0), k_1 = Dec(q \oplus r_1)$，并生成一个随机比特 $s \in \{0,1\}$，计算 $M_0 = m_0 \oplus k_s, M_1 = m_1 \oplus k_{1-s}$，并将 M_0, M_1 发送给接收方。
4. 接收方计算 $m_{c\oplus s} = M_{c\oplus s} \oplus k$。

正确性：$M_{c\oplus s} \oplus k = m_{c\oplus s} \oplus k_c \oplus k = m_{c\oplus s} \oplus Dec(Enc(k) \oplus r_c \oplus r_c) \oplus k = m_{c\oplus s} \oplus k \oplus k = m_{c\oplus s}$。

解释：协议中有很多异或操作看着非常绕，实际上这些异或操作的目的只是为了混淆。

在第3步中，发送方生成随机比特 s 的目的只是为了**混淆消息选择**，等同于把消息 m_0, m_1 随机**打乱顺序后发送**给接收方。因为第2步接收方拥有一个随机比特 c，如果发送方不进行混淆，那么 $M_0 = m_0 \oplus k_0, M_1 = m_1 \oplus k_1$，而接收方拥有 $k = k_c$ 就可以**对应获取**到 m_c（这里的 k 充当一种对称加密的密钥），这意味着接收方最终能获取到哪个消息**完全由他自己生成的随机比特 c 决定**，一旦接收方不遵循协议随机生成 c，就无法保证接收方不能选择消息这个特性了。

既然发送方会混淆输出，接收方选择随机比特 c 的意义是什么？这是因为接收方最后得到的消息是 $m_{c\oplus s}$，如果接收方不随机生成 c，而是在协议中固定 $c = 0$ 或 $c = 1$，那么**发送方就能知道接收方**收到了哪一个消息了，这也不符合OT的要求。同样的，接收方生成随机串 k 也起到了保护 c 这个关键信息的作用。由于 c 和 s 均为随机生成的，因此接收方收到这两条消息的概率分别为1/2。

协议能够保证接收方不了解另一条消息的内容。不妨设 $c = s = 0$，接收方会收到 m_0，对于消息 m_1，由于 $M_1 = m_1 \oplus k_1$，而 $k_1 = Dec(q \oplus r_1) = Dec(Enc(k) \oplus r_0 \oplus r_1)$ 是无意义的随机串，因次接收方无法获得 m_1 的信息。

2. 选择2选1OT协议

我们沿用2选1OT协议的一些假设（消息长度、加解密符号等），可以观察到2选1OT协议中的**两个随机比特 c 和 s 决定了接收方最终收到的是哪一条消息**。如果我们要实现让接收方具有选择消息的能力，只需要**让接收方输入 c，并去掉随机比特 s 的生成**即可。具体协议流程如下：

- **输入**：发送方提供消息 m_0, m_1，接收方提供想要获取的消息索引 $c \in \{0,1\}$
  输出：接收方获得消息 m_c

1. 发送方生成两个随机串 r_0, r_1 并发送给接收方。
2. 接收方生成随机串 k，计算 $q = Enc(k) \oplus r_c$，并将 q 给发送方。
3. 发送方计算 $k_0 = Dec(q \oplus r_0), k_1 = Dec(q \oplus r_1)$，随后计算 $M_0 = m_0 \oplus k_0, M_1 = m_1 \oplus k_1$，并将 M_0, M_1 发送给接收方。
4. 接收方计算 $m_c = M_c \oplus k$。

正确性与相关分析可参考2选1OT协议。

3. n选1OT协议

可以观察到，上述两个2选1OT协议均可自然扩展至对应的n选1OT协议，只许让发送方生成 n 个随机串，c 和 s 的取值范围从 $\{0,1\}$ 变为 $\{0,1,\cdots,n-1\}$ 即可，其余流程基本一致。

四、随机OT 及 预处理优化

Beaver 提出可以把OT拆分为预处理阶段和在线阶段，其中预处理阶段使用的就是现在所说的随机OT协议。我们先介绍随机OT协议，再对 Rabin-OT、2选1OT 和 选择2选1OT 的优化进行介绍。

1. 随机OT协议

该协议的功能类似与上面提到的2选1OT协议，只不过**发送方不再提供输入**，因此我们也只需对应的把发送方的输入**换成协议内随机生成**即可，协议流程如下：

- **输入**：无
  输出：发送方获得随机消息 m_0, m_1，接收方获得消息 m_0 **或** m_1 （概率同等）

1. 发送方生成两个随机消息 m_0, m_1，以及两个随机串 r_0, r_1 并将 r_0, r_1 发送给接收方。
2. 接收方生成随机串 k，以及一个随机比特 $c \in \{0,1\}$，计算 $q = Enc(k) \oplus r_c$，并将 q 给发送方。
3. 发送方计算 $k_0 = Dec(q \oplus r_0), k_1 = Dec(q \oplus r_1)$，并生成一个随机比特 $s \in \{0,1\}$，计算 $M_0 = m_0 \oplus k_s, M_1 = m_1 \oplus k_{1-s}$，并将 M_0, M_1 发送给接收方。
4. 接收方计算 $m_{c\oplus s} = M_{c\oplus s} \oplus k$。

上述流程与2选1OT协议的唯一区别在于第一步中**发送方的两个消息是随机生成**的，其余均可参考2选1OT协议。

2. OT的预处理优化

Beaver 在论文中给出了Rabin-OT、2选1OT 和 选择2选1OT 的优化[3]，本文不对 Rabin-OT 的预处理优化作介绍，因为 Rabin-OT 的实用性有限，而通过 2选1OT 可以轻易构造出 Rabin-OT。对 Rabin-OT 的优化感兴趣的可以自行阅读论文[3]。

通过 2选1OT 构造 Rabin-OT：只需将2选1OT中发送方的其中一个消息 m_0 设置为Rabin-OT中发送方的消息 m，把另一个消息 m_1 设置为公开信息或**无用信息**即可。

预处理思想：Beaver 提出的预处理优化的主要思想是把协议分为预处理阶段和在线阶段。在预处理阶段中发送方和接收方可以**任意调用各种OT协议**来辅助生成信息（预处理**可以提前进行**，不关心这个阶段产生的时间花销），而在线阶段可以利用这些辅助信息加速OT的实现。没错，可以理解为**用OT实现OT**。

2.1 2选1OT协议的预处理优化

预处理阶段

调用一次随机OT，发送方获得两个随机串 r_0, r_1，接收方获得其中一个随机串 r_d，且接收方知道 d 的值。

在线阶段

- **输入**：发送方提供消息 m_0, m_1
  输出：接收方获得消息 m_0 **或** m_1 （概率同等）

1. 发送方生成一个随机比特 $c \in \{0,1\}$，计算 $M_0 = m_0 \oplus r_c, M_1 = m_1 \oplus r_{1-c}$，并将 M_0, M_1 和 c 都发送给接收方。
2. 接收方计算 $m_{c\oplus d} = M_{c\oplus d} \oplus r_d$。

可以看到经过预处理阶段的优化后，在线阶段省去了费时的公钥加解密运算，只需要进行**廉价的异或操作**（或其他对称加密操作），即可实现2选1OT。

分析：正确性类似于未优化的2选1协议，第一步生成的 c 的作用类似于之前的 s，都是为了**混淆消息**，不能仅由接收方自己就能决定收到哪条消息。可能存在疑问：接收方的 d 也是预处理阶段随机生成的，算不算已经混淆过？实际上2选1OT应该达到**在执行协议前无法确定**最终接收方能收到哪条消息，预处理相当于提前计算，真正调用协议是从在线阶段算起的，因此还是需要发送方进行一次混淆。

2.2 选择2选1OT协议的预处理优化

预处理阶段

调用一次随机OT，发送方获得两个随机串 r_0, r_1，接收方获得其中一个随机串 r_d，且接收方知道 d 的值。

在线阶段

- **输入**：发送方提供消息 m_0, m_1，接收方提供想要获取的消息索引 $c \in \{0,1\}$
  输出：接收方获得消息 m_c

1. 接收方计算 $e = c \oplus d$ 并发送给发送方。

2. 发送方计算 $M_0 = m_0 \oplus r_e, M_1 = m_1 \oplus r_{1-e}$ 并发送给接收方。
3. 接收方计算 $m_c = M_c \oplus r_d$。

分析：此处 e 存在的目的就是让接收方想要的消息 m_c 一定要和 r_d 进行异或形成 M_c，因为接收方**只拥有 r_d，所以只能解密和 r_d 异或产生的密文**。我们可以分类讨论：

若 $c=0$，则$e=d$，$M_c = M_0 = m_0 \oplus r_e = m_0 \oplus r_d$，接收方可以解密获得 m_0。
若 $c=1$，则$e=1-d$，$M_c = M_1 = m_1 \oplus r_{1-e} = m_1 \oplus r_d$，接收方可以解密获得 m_1。

由于接收方不知道 r_{1-d}，因此**无法获取另一条消息 m_{1-c}** 的内容。发送方不知道 c 和 d，因此也**无法了解到接收方收到了哪一条消息**。

参考文献

- [1] M. Rabin. "How to Exchange Secrets by Oblivious Transfer." TR-81, Harvard, 1981.
- [2] S. Even, O. Goldreich, A. Lempel. "A Randomized Protocol for Signing Contracts." Proceedings of Crypto 1982 , Springer-Verlag, 1983, 205-210.
- [3] Beaver, D. Precomputing Oblivious Transfer. In: Coppersmith, D. (eds) Advances in Cryptology — CRYPT0' 95. CRYPTO 1995. Lecture Notes in Computer Science, vol 963. Springer, Berlin, Heidelberg.

本文为作者在学习相关知识时的一种记录，便于以后的回顾。作者并没有系统地学习过密码学，因此在表述上可能会存在不严谨甚至出错的地方，文章仅供参考，欢迎大家与我交流，一起进步！

其他平台：

- 知乎（Totoro）：https://www.zhihu.com/people/totoro-14-60
- CSDN（_Totoro_）：https://blog.csdn.net/orz_Totoro
- B站（Totoro_134）：https://space.bilibili.com/279377771
- Github（Totoro134）：https://github.com/Totoro134
- 公众号（知识长生所）