不经意传输：隐私计算中的信息传递组件

July 19, 2023

摘要

在本文中，我们将对不经意传输（OT）这一在隐私计算中起到至关重要作用的密码学组件进行概述，讨论范围包括其产生原因、基本原理、发展过程、应用场景和发展方向，并通过引入半诚信和恶意敌手模型探讨不同的 OT 算法的安全性。我们对 OT 作了正式定义，通过一个典型的构造方案进行引入，并在概述该领域发展过程中插入不同类型的研究成果，例如 OT extension 实现的高效快速信息交换等，并通过一些实际案例说明这一密码学组件的应用场景，从这些应用中我们也能够看出 OT 的研究方向。最后，在对不经意传输本质和领域发展方向进行一些思考时，对于 OT 仍旧存在的一些开放问题给出一些思考与个人见解。

1 从多方安全计算：混淆电路讲起

设想以下在机器学习语境中很有可能会发生的场景：某搜索引擎 A 和搜索引擎 B 拥有各自用户的浏览偏好数据集，两网站希望通过一同训练某个模型来实现对于用户画像的更精确刻画。一个很自然的解决方案是将双方的数据集进行修剪和合并以融合成一个更大的数据集，这个整体很显然的会在训练精度和稳定性方面胜过小数据集。

问题在于在考虑到隐私保护的要求和数据作为资产的性质，这一方案并不容易实现。因此诞生了隐私计算的概念，即要求双方在互不知悉对方输入的情况下共同计算某一函数并得到结果。这一想法最早的来源是姚期智提出的百万富翁问题。而作为应对方案，姚提出了混淆电路的概念，不经意传输作为最终双方求解过程中一个必不可少的组成部分，对于姚氏电路的最终实现起到了至关重要的作用。我们将原方案中相关的片段进行如下提取：

富翁 A 生成了一个原始明文 $\{0,1\} \to \{x_0, x_1\}$ 的对照表，若需要成功解密，则 B 需要并且只能获得自己所持有的 i 值对应的 x_i，若无法获得则解密无法进行；若 B 获得则整张表则可以倒推 A 持有的 j 值的信息；若 A 知晓 B 查询了哪个 i 对应值则侵犯了 B 的隐私。这一难题就是最典型的不经意传输应用场景。

以下是对不经意传输的正式定义：不经意传输（oblivious transfer）是一个密码学协议，在这个协议中，消息发送者从一些待发送的消息中发送一条给接收者，但事后对发送了哪一条消息仍然 oblivious（不知晓），这个协议也叫茫然传输协议。习惯上，我们会把从 n 条消息中不经意传输 k 条的方案称为 k-out-n OT，记作 OT_n^k。当 k 为 1 时通常简记为 OT^n。

OT 协议是通信双方用来传递秘密信息的协议，它是密码学理论中最基本、最重要的原语之一，早期多用于构建公平秘密交换协议、抛币协议、公平电子合同签署协议、零知识证明协议等重要传输方案。而其当下最热门的应用是作为安全多方计算协议中大量使用的关键构建模块. 早在

1988 年,Kilian 便提出了一个著名的结论: 拥有一个实现不经意传输的黑盒就可以完备地构建任何一个安全计算协议. 其基本性质保证了传输的保密性等一系列要求，是安全多方计算的一大基石.

本文的行文结构如下：首先尝试从一个简单易懂的 OT 方案讲起，并选取一些典型的例子讲解从 1-out-2 OT 到 k-out-n OT 以及效率更高的 OT 扩展方案所使用的思路。接着，我们会探讨半诚信模型下与恶意模型下对于 OT 协议提出的不同要求，并涉及一个能够抵抗恶意一方的传输方案。接着，我们回顾一些经典 MPC 协议中使用到的 OT 场景，以此展示其在构造安全多方计算协议中的重要作用。最后，我们对 OT 协议本质以及当下研究方向进行一些总结。

2 一个典型构造方案

在这个部分将构造一个简单易懂且十分实用的交互方案来帮助理解 OT 协议需要做到什么以及如何达成这些目的。我们先来看看在两方情况下 OT 模型是如何定义的。

2.1 典型 OT 模型定义

我们假设 Alice 有两个数值 v_0 和 v_1，Bob 想知道其中的一个 $v_i, i \in \{0,1\}$。通过执行 OT 协议，Bob 知道了 v_i，但不知道 v_{1-i}；同时，Alice 不知道 i。总而言之，一个标准的不经意传输协议应当达到以下三个目标:

- 接收者得到 v_i 的值。
- 发送者不知道 i 的值。
- 接收者不知道 v_{1-i} 的值。

为了表述方便，目前我们考虑的是接收者从发送者的两项数据中选择一项，即 1-out-2 OT。之后会谈到如何将 1-out-2 OT 扩展到 1-out-n OT，即接收者从发送者的 n 项数据中选择一项。

2.2 一个简单的 1-out-2 OT 协议构造

预设

1. Alice 拥有值 v_0, v_1 和密匙 s, r_0, r_1.
2. Bob 拥有值 $i \in \{0,1\}$ 和密匙 k,Bob 想获得 v_i.
3. Alice 和 Bob 事先统一 $g \in Z_p$，其中 g 是大整数，p 是大素数.

协议流程

1. $Alice \to Bob$: $Alice$ 选取 s 并向 Bob 发送 g^s.
2. Bob 根据自己想要的消息基于 i 生成 $L_i = \begin{cases} g^k & \text{if } i = 0 \\ g^{s-k} & \text{if } i = 1 \end{cases}$
3. $Bob \to Alice$: Bob 向 $Alice$ 发送 L_i.
4. $Alice$ 生成 C_0, C_1.
   $C_0 = (g^{r_0}, (L_i)^{r_0} \oplus v_0)$ 其中 $\oplus$ 表示逐位异或，下同.
   $C_1 = (g^{r_1}, (g^s / L_i)^{r_1} \oplus v_1)$
5. $Alice \to Bob$: $Alice$ 向 Bob 发送 C_0, C_1.
6. Bob 解密 v_i
   1.case i=0

Bob 可以通过如下方式解密获得 v_0:

$C_0[0]^k \oplus C_0[1] = (g^{r_0})^k \oplus (L_i)^{r_0} \oplus v_0 = (g^{r_0})^k \oplus \left(g^k\right)^{r_0} \oplus v0 = v0$

2.case i=1

类似地，Bob 通过如下方式解密获得 v_1:

$C_1[0]^k \oplus C_1[1] = (g^{r1})^k \oplus (g^s/L_i)^{r_1} \oplus v_1 = (g^{r_1})^k \oplus \left(g^k\right)^{r_1} \oplus v_1 = v_1$

思路分析

离散对数的数学难题是在此过程中保证单向性的基础，第一步中 Alice 选择的 s 对 Bob 完全不可知，此处 Bob 即便知道 g 和 g^s 也无法破译 s, 因为离散对数问题不存在高效解法.

第二步中，在 Bob 选择 i 并生成 L_i 时，即便发送给 Alice 的是 g^k，同上推导可知 Alice 同样不能推导出 Bob 所选择的 k. 同时，由于 Alice 对于第二步中 Bob 选择的 k 一无所知，所以 g^k 和 g^{s-k} 对于 Alice 完全均匀，因此 Alice 无法知道 Bob 发来的是 g^k 还是 g^{s-k}，更无法知道 Bob 选择的 i. 不经意传输中最基础的“发送方不知道接收方要哪些数据”要求在这一步得到了满足.

第三步的作用是保证 Bob 除了自己想要的 v_i 之外无法再获得另外一个值，我们来看 Alice 的构造方法是如何保证这点的：首先，Bob 知道 g, g^{r_0}, g^{r_1} 也无法破译 r_0, r_1, 因为离散对数问题不存在高效解法；而在解密过程中，若 Bob 发送的是 L_0，则无法获得 v_1, 因为:

$$C_1[1] = (g^s/L_i)^{r_1} \oplus v_1 = g^{(s-k)r1} \oplus v_1$$

而 Bob 不知道 s, r_1. Bob 能获得的知识包括 k、g^s、g^{s-k}，想要获得 $C_1[1]$ 就必须计算出 $g^{(s-k)r1}$，因而困难性归结为从 g^{r_1} 求解 r_1，这步的安全性由离散对数求解困难性保证.

同样的，若 Bob 发送的是 L_1，则 Bob 无法获得 v_0, 因为:

$$C_0[1] = (L_i)^{r0} \oplus v_0 = g^{(s-k)r0} \oplus v_0$$

而 Bob 不知道 s, r_0. 同上，Bob 能获得的知识包括 k、g^s、g^{s-k}，想要获得 $C_0[0]$ 就必须计算出 $g^{(s-k)r0}$，因而困难性归结为从 g^{r_0} 求解 r_0，这步的安全性也由离散对数求解困难性保证，和上文 i=0 的情况恰好是对偶的.

因此，Bob 只能解密自己选择的 i 对应的 v_i 而不能解密 v_{1-i}. 这一设计保证了 Bob 不多获得信息，从而保护了 Alice 的隐私. 可以看出，这个协议的安全性设计均基于离散对数困难问题.

3 历史与发展

这个部分对 OT 发展历史中出现的典型方案进行概述，其中 k-out-n OT 与 IKNP 方案具有很高的实用价值，因此我们花较大篇幅对其进行底层原理的详细讲解。严格来说 OT 协议及衍生分类远不止这些，我们会在这部分结尾进行简短的列举，但其基本都是以下这些概念的拓展。按照叙述的顺序我们可以看到从起步阶段不保证成功传输的协议，到 1-out-of-2-OT、1-out-of-n-OT，再到后来的 k-out-of-n-OT，传输的效率是一个不断提高的过程。而 k-out-of-n-OT 和下一小节提到的 OT extension 又是不同的发展方向，后者实现了多个 OT 协议之间进行扩展。

3.1 起步阶段

3.1.1 提一次提出：Rabin.1981

不经意传输 (Oblivious Transfer - OT) 最早在 1981 年被 Michael O. Rabin 提出，它是为了解决如下的问题而产生：Alice 拥有秘密 S_A，Bob 拥有秘密 S_B. Alice 和 Bob 想要交换秘密，要

求两方都有可能得到秘密并且秘密拥有方不知道对方是否得到秘密。具体方案如下：

预设

两方的秘密都是单比特.

交换流程

1. Alice 随机选取两个大素数 p、q. 并计算得到 $one-time-key n_A$，然后将 n_A 发送给 Bob.

2. Bob 随机选取一个数 x，要求 $x < n_A$，计算 $c \equiv x^2 \bmod n_A$，然后将 c 和私钥加密的 x 发送给 Alice。

3. Alice 找到一个 x_1 使得 $x_1^2 \equiv c \bmod (n_A)$，发送 x_1 给 Bob.

4. Bob 计算 $\gcd(x - x_1, n_A) = d$，此时有 $p(d == qord == q) = \frac{1}{2}$.

5. Bob 根据下面公式计算 v_B:

$$v_B = \begin{cases} 0, & \text{if } (x - x_1, n_A) = p \text{ or } q \\ 1, & \text{otherwise} \end{cases}$$

接着计算 $\varepsilon_B = v_B \oplus S_B$, 然后将 ε_B 发送给 Alice.

分析

以上是 Alice 获得 Bob 的秘密 S_B 的过程，可以得出 Alice 得到 S_B 的概率为 1/2. Bob 获得 S_A 的过程依然是上述步骤，只不过是将 Alice 和 Bob 角色互换.

Rabin 提出的方案两方都无法获得对方的秘密的概率是 1/4, 可以成功交换的概率是 3/4. 可见该方案还不是很完善，不能保证两方每次都能在满足要求的情况下获得秘密，因此不具有应用意义. 所以有了 1985 年 Even 等人在此基础上提出的新的 1-out-2 OT 协议.

3.1.2 Even.1985:1-out-of-2 OT

Even 等人的提出新的使用公钥密码体制的 1-out-of-2 OT 协议，给出了 OT 公理化的定义和实现。相比于 Rabin 等人提出的一方只有 1/2 的概率获得秘密，Even 等人将其进行了改进，即：Alice 拥有两个秘密 (M_0, M_1)，而 Bob 想要知道其中一个. 在 OT 协议执行完成之后，Bob 获得了其中一个秘密，但是不知道另外一条秘密，并且 Alice 也不知道 Bob 选择的是 M_0 还是 M_1. 这一个方案和我们在第二部分所描述的 OT 在最终功能上是一样的，但是实现方式略有不同，接下来我们简略描述一下.

传输过程如下图：

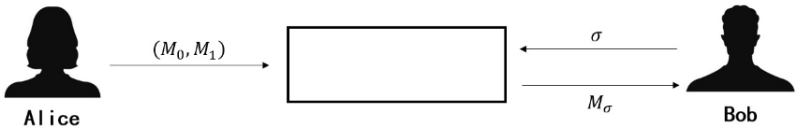

Figure 1: 1-out-of-2 OT

交换流程

1. Alice 在公钥密码中随机选择一组密钥 (pk, sk), 同时随机选择加密算 32 法明文空间 R 的两个明文 m_0, m_1, 最后将 E_{pk}, m_0 和 m_1 发送给 Bob.

2. Bob 随机选择 $r \in \{0, 1\}$, 然后在与步骤 1 中同样的明文空间选择 k; 计算 $q = E_{pk}(k) \boxplus m_r$, 并且向 Alice 发送 q.

3. Alice 计算 $k'_i = D_{sk}(q \boxminus m_i)$, 其中 $i \in \{0,1\}$; Alice 随机选择变量 $s \in \{0,1\}$; 进而将三元组 $(M_0 \boxplus k'_s, M_1 \boxplus k'_{1-s}, s)$ 发送给 B o b .

4. Bob 根据自己选择 r 以及三元组最后一项 s 来选择第一项或者第二项; $\boxplus$ 是模运算加法, $\boxminus$ 是模运算减法。模运算的 n 就是明文空间大小.

可以清楚的看到，1-out-of-2 OT 执行结束之后，Bob 获得了一个秘密且不知到另外一条秘密，而 Alice 则不知到 Bob 拿到了哪一条秘密。1-out-of-2 OT 是一个具有实际应用意义的不经意传输协议，也是目前较为常用的一种。

3.1.3 Brassard.1986:1-out-of-n OT

之后在 1986 年, Brassard 等人继续将 OT 协议改进到了 1-out-of-n OT 版本。与上述 1-out-2 OT 中的问题基本相同，唯一变化的就是从 2 条秘密传递 1 条给 Bob 变成了从 n 条秘密传递给 Bob.

初始协议

n 条秘密为: $M_1, M_2, \ldots, M_n$, 且每个消息长为 t bits，设 $b_{i,j}$ 表示第 i 个秘密的第 j 位，可见 $b_{i,j} \in \{0,1\}$.

1. 首先 Alice 随机选择大素数 p,q ，计算 $m = pq$，并计算模 m 的二次非剩余 y，然后对于每一个比特位 $b_{i,j}$ 选择一个整数 $x_{i,j}$ 并计算 $z_{i,j} = x_{i,j}^2 y^{b_{i,j}} \bmod m$ （显然，当且仅当 $b_{i,j} = 0$ 时 $z_{i,j}$ 时二次剩余) 然后将 $m, y, z_{i,j}$ 发送给 Bob。

2. Bob 选择随机数 r 以及随机的比特位 $\alpha \in 0,1$，计算 $q = z_{i,j} r^2 y^{\alpha} \bmod m$，其中当且仅当 $\alpha = b_{i,j}$ 时，q 是 m 的二次剩余. 如此一来，只需要验证 (2) 中得到的 q 是否为 m 的二次剩余即可，若是则 $\alpha = b_{i,j}$ 反之则不等.

每一轮上述过程 Bob 可以得某个消息的一个比特位，重复 t 次，即可得到秘密 M_i .

缺点分析

虽然上述方法可以完成目标，但是仍存在三个缺点:

1. Bob 可能询问的是不同秘密的比特位。

2. Bob 可能得到两个消息之间的异或。

3. Alice 可能欺骗 Bob，发送的 y 可能是一个二次剩余，也就有可能指出 Bob 的选择。

为克服以上缺点，Brassard 引入一个优化的方案. 在这一小节我们看到原来的协议很多都是基于全诚信或者半诚信的条件，这就导致了某些协议在实际运用中很有可能会被一方通过某种恶意选值方法获取到额外的信息，这也对模型提出了更强的安全要求，这个细节我们会在 3.3 节涉及，下面我们先来看优化后的 1-out-n OT 方案.

优化协议

1. Bob 随机选择扰动函数 $\varepsilon()$，随机整数 $r_{k,j}$ 和随机比特位 $\alpha_{k,j} \in \{0,1\}$，使得 $k = \varepsilon(i)$，其中 i 表示 Bob 想要获得第 i 条秘密。并计算 $q_{k,j} = z_{i,j} r_{k,j} y^{\alpha_{k,j}} \bmod m$ 后将 t 个 $q_{k,j}$ 传送给 Alice，同时需要向 Alice 证明所有的 $q_{k,j}$ 可用性（一个理解是以此来保证请求的是同一条消息的不同比特位，也就是解决缺点 1 和 2).

2. Bob 传送 k 给 Alice($k = \varepsilon(i)$).

3. Alice 对于每一个 $q_{k,j}$ 都给出模 m 下的二次特征值，发送给 Bob。

4. Bob 根据二次特征值推测出相应比特的数据（特征值为零则与 α 不同，不为零则相同).

这便是最开始的 1-out-of-n OT 协议。至此最基本的 OT 协议包括 2 取 1，n 取 1 都已经实现.

3.1.4 k-out-of-n OT*

在上面的讨论过程中我们已经实现了可供选择的总消息数从 1 到 n 的转变，但是在这个过程中收方依然只能够每次运行 OT 协议获得一条消息，那么，有没有可能一次从可供选择的多条消息中不经意地传输 k 条呢？接下来这个方案巧妙地利用多项式的根实现了 k-out-n OT 传输。

为了深入理解这个协议的精妙之处，还是先来看一个一元多项式 $f(x) = b_0 + b_1x + \cdots + b_{k-1}x^{k-1} + x^k \equiv (x-\sigma_1)(x-\sigma_2)\cdots(x-\sigma_k) \bmod q$: 我们很轻易地就能知道在 σ_i 处函数 $f(x)$ 取值为 0. 那么如果存在一个第三者对函数 $f(x)$ 乘上一个常系数 k 之后再对 h 取指数得到 $h^{kf(x)}$，由于 k 的加入和离散对数求解的困难性，我们对于 $h^{kf(x)}$ 将无法求解 k 以及一系列相关信息，但是，我们对于这个函数能够利用到一些很特殊的性质，即：当 x 取到 σ_i 时 f(x) 为 0，于是 $h^{kf(x)}$ 指数项为 0，从而整个式子值为 1；而除了这些根之外，我们就对 $h^{kf(x)}$ 一无所知了，这就是接下来构造协议所基于的的数学困难问题。

预设

- 参数选取: $(g, h, \mathbb{G}_q)$;
- 发送方 S 有 m 条消息:$m_1, m_2, \ldots, m_n$;
- 接收方 R 需要的消息标识符共 k 条:$\sigma_1, \sigma_2, \ldots, \sigma_k$;

协议流程

1.R 随机选取两个多项式 $f(x) = a_0 + a_1x + \cdots + a_{k-1}x^{k-1} + x^k$ 和 $f'(x) = b_0 + b_1x + \cdots + b_{k-1}x^{k-1} + x^k$. 上式满足 $a_0, a_1, \ldots, a_{k-1} \in_R \mathbb{Z}_q and b_0 + b_1x + \cdots + b_{k-1}x^{k-1} + x^k \equiv (x-\sigma_1)(x-\sigma_2)\cdots(x-\sigma_k) \bmod q$.

2. $R \longrightarrow S : A_0 = g^{a_0}h^{b_0}, A_1 = g^{a_1}h^{b_1}, \ldots, A_{k-1} = g^{a_{k-1}}h^{b_{k-1}}$.

3. S 计算 $c_i = \left(g^{k_i}, m_i B_i^{k_i}\right)$

上式满足 $k_i \in_R \mathbb{Z}_q$ 和 $B_i = g^{f(i)}h^{f'(i)} = A_0A_1^i \cdots A_{k-1}^{i^{k-1}}(gh)^{i^k} \bmod p,\ i = 1, 2, \ldots, n$.

4. $S \longrightarrow R : c_1, c_2, \ldots, c_n$.

5. 令 $c_i = (U_i, V_i)$. R 对每个 σ_i 计算 $m_{\sigma_i} = V_{\sigma_i}/U_{\sigma_i}^{f(\sigma_i)} \bmod p$.

思路分析

在上面这个交互过程中，接收方 R 所持有的 $f'(x)$ 就是我们在这小节的引言部分所讲到的能够分解为 k 个一次式乘积的函数，$f(x)$ 则是作为混淆项使得 S 方想要从 $A_i = g^{a_i}h^{b_i}$ 推导出多项式系数 b_i 非常困难，这就使得发送方无从得知 R 想要哪些元素。

另一方面，发送方 S 所做的事情是取一个 k_i 以实现对于 $B_i = g^{f(i)}h^{f'(i)}$ 的一个扰乱，使得 R 对于 B_i^k 的性质除了该式在 $f'(x)$ 的根 σ_i 处取值为 $g^{k_i f(\sigma_i)}$ 之外一无所知，从而 R 不可能从 S 发过来的信息中推导出其它的记录，这也就保证了“接收方不多获得信息”的要求。以上便是这个 k-out-n 方案的设计原理.

正确性的数学语言描述如下，结合上面两段分析更容易理解。

对于 B_i 有

$$\begin{aligned} B_i &= A_0A_1^i \cdots A_{k-1}^{i^{k-1}}(gh)^{i^k} \\ &= g^{a_0+a_1i+\cdots+a_{k-1}i^{k-1}+i^k}h^{b_0+b_1i+\cdots+b_{k-1}i^{k-1}+i^k} \\ &= g^{f(i)}h^{f'(i)} \bmod p \end{aligned}$$

对于 $c_i = \{U_i, V_i\}$，接收方计算

$$\begin{aligned} m'_{\sigma_i} &= V_{\sigma_i} / U_{\sigma_i}^{f(\sigma_i)} = m_{\sigma_i} B_i^{k_i} / \left(g^{k_i f(\sigma_i)}\right) \\ &= m_{\sigma_i} g^{k_i f(i)} h^{k_i f'(i)} / g^{k_i f(\sigma_i)} \end{aligned}$$

由于 σ_i 为 $f'(i) = 0$ 的解之一，因此有

$$\begin{aligned} m'_{\sigma_i} &= m_{\sigma_i} g^{k_i f(i)} h^{k_i f'(i)} / g^{k_i f(\sigma_i)} \\ &= m_{\sigma_i} g^{k_i f(i)} / g^{k_i f(\sigma_i)} \\ &= m_{\sigma_i}. \end{aligned}$$

3.2 不经意传输扩展及优化

我们先从公钥加密 (PKE) 说起, 由于公钥加密的开销比对称加密 (SKE) 的开销大, 因此我们通常通过同时使用 PKE 和 SKE 的混合加密来最小化 PKE 的开销: 首先使用昂贵的 PKE 加密短密钥 s, 然后使用使用 s 通过廉价的 SKE 来加密长信息 M. 于是, 通过 λ 比特的 PKE 和廉价的 SKE, 我们实现了 N 比特的 PKE, 其中 λ 代表计算安全参数. 那么, 有没有类似于混合加密的方法, 仅通过 λ 次 OT 实例和廉价的 SKE 来得到大量的 OT 实例呢? 这就是 OT 扩展的出发点.

在这个部分我们会涉及两个协议：Beaver OT extension 和 IKNP，前者不具备现实意义但是提供了理论可行性，因此不进行深入探讨；IKNP 协议则非常完美的实现了前者所提出的一系列想法，其巧妙地利用了异或的特性来实现不经意传输扩展，我们会在相应的小节详细讨论。

3.2.1 Beaver OT extension

Beaver 在 1996 年提出了 OT 扩展 (OT extension) 的概念. Beaver 注意到 Yao 协议需要的 OT 数量正比于函数的输入的长度, 我们可以使用混淆电路来构造这样一个 OT 扩展协议: Alice 和 Bob 分别输入长度为 λ 比特的 s_A 和 s_B，使用 $s_A \oplus s_B$ 作为伪随机种子:

(1) 生成 2n 个随机串，组成 n 对随机消息 $(m_{1,0}, m_{1,1}), \cdots, (m_{n,0}, m_{n,1})$ ；

(2) 选取一个 n 比特串 r；

最终 Alice 得到所有随机消息对 $\{m_{i,b}\}_{i,b}$，Bob 得到 r 以及 r 的第 i 比特 r_i 所对应的消息 $\{m_{i,r_i}\}_i$，其中 $i \in \{1, \cdots, n\}$，这里的 $n \gg \lambda$.

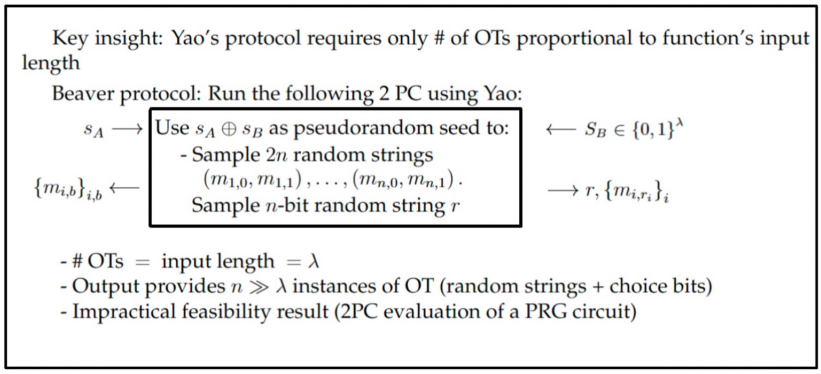

Figure 2: Beaver OT extension

通过这种方式生成的 ROT 实例数为 n，而实际使用的 ROT 实例数仅为 λ，通常将这部分实际使用的 OT 实例称为 Base OT. 虽然 Beaver 的 OT 扩展协议并不实用，因为需要使用 Yao 协议对包含大量 PRG 的布尔电路求值, 但它告诉我们 OT 扩展的想法是切实可行的.

3.2.2 IKNP：高效的 OT 扩展协议 *

2003 年的美密会 Yuval Ishai, Joe Kilian, Kobbi Nissim, Erez Petrank 共同发表了“Extending Oblivious Transfers Efficiently”，即著名的 IKNP 协议, 其原理主要是基于矩阵变换, IKNP 协议标志着 OT 扩展逐渐走向实用. 接下来详细解释其作用原理.

假设发送方手中有 n 对数据 $(x_0^1, x_1^1; x_0^1, x_1^2; \ldots x_0^n, x_1^n)$，接收方有 n 个选择位 $(i_0, i_1, \ldots, i_n)$，接收方希望从发送方手中获得 n 个选择位所标识的数据。一个直观且简单粗暴的办法是直接执行 n 次 1-out-2 不经意传输协议，但这样的实现方式会使得通信复杂度随着 n 增加而增加，传输开销和计算开销也会增长到一个令人无法忍受的程度。而不经意传输扩展协议只需要执行 k 次不经意传输即可达到和执行 n 次不经意传输同样的效果，因而能够极大地提升不经意传输的适用性，这里需要着重强调的是，上文的 k 为固定值，与 n 的大小无关。

预设

发送方 S 的输入为 n 对数据 $(x_0^1, x_1^1; x_0^1, x_1^2; \ldots x_0^n, x_1^n)$

接收方 R 的输入为 n 个选择位，$(i_0, i_1, \ldots, i_n), i \in \{0, 1\}$

双方共同持有一个安全参数 k（即执行不经意传输的次数）

假设存在一个随机原语（通常是一个哈希函数）

假设存在一个 1-out-2 的不经意传输（根据前文的构造这是很容易实现的）

协议流程

1. 发送方 S 初始化一个长度为 k 的随机向量 $s \in \{0,1\}^k$，接收方 R 初始化一个大小为 $n \times k$ 的随机 bit 矩阵 T。

2. 双方执行 k 次 1-out-2 不经意传输。在这 k 次不经意传输中，S 作为接收方，持有 k 个选择位，即 $s_0, s_1, \ldots, s_k$，R 作为发送方，持有 k 对数据，$(t^a, i \oplus t^a), 1 \le a \le k$，其中 t^a 表示 T 的第 a 列，i 为 R 的 n 个选择位组成的向量。

3. 令 Q 为发送方 S 在第 2 步的 k 次不经意传输中收到的数据组成的 $n \times k$ 矩阵，Q 的每一行和每一列满足以下性质：$\mathbf{q}^a = (\mathbf{s}_a \& \mathbf{i}) \oplus \mathbf{t}^a, \mathbf{q}_b = (\mathbf{i}_b \& \mathbf{s}) \oplus \mathbf{t}_b$，其中 q^a 表示 Q 的第 a 列，q_b 表示 Q 的第 b 行。随后，发送方 S 发送 n 个数据对 $\left(y_0^b, y_1^b\right), 1 \le b \le n$, 其中 $y_0^b = x_0^b \oplus H(b, \mathbf{q}_b), y_1^b = x_1^b \oplus H(b, \mathbf{q}_b \oplus \mathbf{s})$。

4. 接收方计算 $H(b, t_b)$，对 $y_{i_b}^b$ 得到 $x_{i_b}^b$。

思路分析

理解上述协议的关键之处在于理解第三步，阅读原文后结合本人理解作图如下，或许能帮助快速厘清实现原理。右侧的蓝色矩阵是接收方 R 随机初始化的一个向量，$1 \times n$ 的列向量是 R 最终想要想要获得的 n 个数据的对应标识位 (0 代表想要第一个,1 代表想要第二个)。

那么，在执行 k 次预 OT 传输后，S 拿到的矩阵就是 R 的原始 $n \times k$ 矩阵与在 S 初始化的 k 个元素行向量值为 1 的位置对应的列叠加上 R 的选择位列向量，如左侧矩阵所示。

单独抽选一行来看, 就能发现这种操作的独特性质: 若 R 对应的选择位为 0, 则该行原矩阵相同，不受影响；若 R 对应的选择位为 1，则该行恰好等于原矩阵的行与 S 初始化的向量进行异或叠加，由于异或的性质这一行再次逐位异或后就又得到原来的行，但由于 S 初始化的向量对于 R

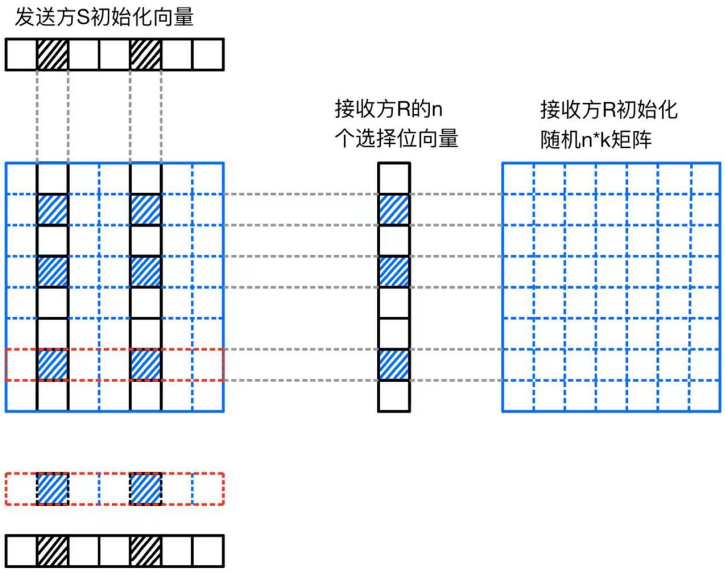

Figure 3: 原理图解

完全不可见，这就导致了 $y_0^b = x_0^b \oplus H\left(b, \mathbf{q}_b\right), y_1^b = x_1^b \oplus H\left(b, \mathbf{q}_b \oplus \mathbf{s}\right)$ 中的 $H\left(b, \mathbf{q}_b\right)$ 和 $H\left(b, \mathbf{q}_b \oplus \mathbf{s}\right)$ 对于发送方而言只能计算出自己选择的位对应的项，从而只能得到选择位的消息.

以上内容的数学语言描述如下：矩阵 $\mathbf{Q}$ 的每一行和每一列满足 $\mathbf{q}^a = (\mathbf{s}_a \& \mathbf{i}) \oplus \mathbf{t}^a, \mathbf{q}_b = (\mathbf{i}_b \& \mathbf{s}) \oplus \mathbf{t}_b$，其中 $\mathbf{q}^a$ 表示 Q 的第 a 列，$\mathbf{q}_b$ 表示 Q 的第 b 行。对于第一项，倘若 $\mathbf{s}_a$ 为 0 ，则 $(\mathbf{s}_a \& \mathbf{i}) = 0^n$，从而 $\mathbf{q}^a = \mathbf{t}^a$；倘若 $\mathbf{s}_a$ 为 1 ，则 $(\mathbf{s}_a \& \mathbf{i}) = \mathbf{i}$，从而 $\mathbf{q}^a = \mathbf{i} \oplus \mathbf{t}^a$，这正好与 k 次不经意传输时 R 发送的 k 对数据是一致的。由第一项，我们可以进一步推理出第二项。由第一项可知: $Q_b^a = (\mathbf{s}_a \& \mathbf{i}_b) \oplus T_b^a$，因而有

$$\mathbf{q}_b = \left(Q_b^1, Q_b^2 \ldots, Q_b^k\right) = \left((\mathbf{s}_1 \& \mathbf{i}_b) \oplus T_b^1, (\mathbf{s}_2 \& \mathbf{i}_b) \oplus T_b^2, \ldots, (\mathbf{s}_k \& \mathbf{i}_b) \oplus T_b^k\right) = (\mathbf{i}_b \& \mathbf{s}) \oplus \mathbf{t}_b$$

从上述式子中我们我们可以进一步推断 $\mathbf{q}_b$ 的性质. 倘若 $\mathbf{i}_b = 0$，则 $\mathbf{q}_b = \mathbf{t}_b$，倘若 $\mathbf{i}_b = 1$ ，则 $\mathbf{q}_b = \mathbf{s} \oplus \mathbf{t}_b$ (推理步骤同上一段。因此，在生成 n 个数据对 $\left(y_0^b, y_1^b\right)$ 时，能够保证 $y_{i_b}^b = x_0^b \oplus H\left(b, \mathbf{t}_b\right)$，从而保证接收方能够成功解密其所需要的数据.

OT 扩展可以被看作是一种对原始 OT 协议的优化和扩展，旨在提高效率。通过利用原始 OT 协议的特性和加密技术，OT 扩展能够同时执行多个 OT 操作，减少了通信和计算的次数，从而提高了整体效率。因此，OT 扩展可以被认为是 OT 的快速实现方式之一，可以提供更高的吞吐量和更好的性能。当然，OT 扩展并不是 OT 的唯一快速实现方式，还有其他方法和技术可以在 OT 中提高效率，如利用硬件加速器、并行计算和优化算法等。

3.2.3 总结与比对

k-out-of-n OT（k-out-of-n Oblivious Transfer）和 OTextension（Oblivious Transfer Extension）是两种不同的扩展方式，用于增强 Oblivious Transfer（OT）协议的功能和效率。它们之间存在一些区别，如下所述：

功能：k-out-of-n OT 是一种扩展方式，用于在单个 OT 协议中进行多个选择。它允许接收方从发送方提供的 n 个选择中选择 k 个，并获得所选选择的值，同时不知道未选择的选择的值。

k-out-of-n OT 扩展了单个 OT 的选择能力，从而提供更强大的功能。OTextension 是一种扩展方式，用于在多个 OT 协议之间进行扩展。它允许在多个 OT 协议之间共享一些公共参数，从而减少计算和通信的开销。OTextension 主要旨在提高 OT 协议的效率和可扩展性。

使用场景：k-out-of-n OT 主要适用于需要进行多个选择的场景，其中接收方需要从多个选择中选择特定数量的选项。例如，在加密投票协议中，选民可以从多个候选人中选择多个选项。OTextension 主要适用于需要进行多次 OT 的场景，其中多个 OT 协议需要同时执行，或需要重复执行多次。通过共享公共参数，OTextension 可以减少多个 OT 协议之间的通信和计算开销，提高整体效率。

实现方式：k-out-of-n OT 的实现方式通常基于特定的密码学构造，如承诺方案（commitment schemes）或秘密分享（secret sharing）。它涉及设计和分析算法来实现从 n 个选择中选择 k 个的功能。OTextension 的实现方式通常涉及使用密码学原语，如伪随机函数（pseudorandom functions）或同态加密（homomorphic encryption）来生成和传输公共参数，以减少重复的计算和通信。它通常涉及协议设计和优化，以提高整体效率。

总而言之，k-out-of-n OT 和 OTextension 是两种不同的扩展方式，它们提供了不同的功能和效率优化。k-out-of-n OT 扩展了单个 OT 的选择能力，而 OTextension 主要用于提高多个 OT 协议的效率和可扩展性。

3.3 恶意安全的 OT 协议

在半诚实敌手和恶意敌手两种模型下对于协议设计提出的要求是完全不同的. 这部分我们会涉及半诚信假设与恶意假设的讲解，并分析一个 IKNP 协议的优化版本，比较其与原协议的不同，借此说明恶意安全的 OT 协议在设计时应该注意什么.

3.3.1 半诚信假设与恶意假设

我们来回顾一下上文的 IKNP 协议. 安全性方面, IKNP 协议是半诚实安全的. 具体而言, Alice 可以是恶意的, 但 Bob 只能是半诚实的, 下面将介绍如果 Bob 是恶意的, IKNP 协议会存在的问题.

假设 Bob 在生成选择比特矩阵时, 在某一行篡改了其中第 i 比特. 则 Alice 和 Bob 在该行 i 位有一比特不一致, 当且仅当 Alice 在在第一步选择的向量 S 的第 i 个比特 $s_i = 1$. 若 Alice 对该行进行 Hash 运算并在其他大型协议中使用这些结果而被 Bob 检测到, 则 Bob 将能得到 s 的一比特信息! Bob 通过使用类似地方法, 在每列的不同位置篡改一个比特, 便可以获得 Alice 在该 OT 扩展协议中唯一的秘密信息 s, 进而攻破整个 OT 扩展协议. 因此, IKNP 协议仅是半诚实安全的.

在大部分协议中，实现发送方不知道接收方的选择是非常容易的，但是收方经常可以通过协议设计中的一些纰漏获得额外的消息，因此，大部分恶意假设都是基于接收方想要获得额外信息因此发动攻击这一假设的，为了阻止这一意图，一些半诚信 OT 协议需要对接收方进行更为严密的限制.

3.3.2 KOS：恶意安全的 IKNP 协议

回顾 IKNP 协议的等式 $q_i = t_i \oplus R_i \wedge s$ 和恶意敌手 Bob 的整个攻击过程, 这是由于 R 的第 i 行 R_i 所有元素不一致造成的. 若 R_i 的汉明重量 (Hamming Weight) 很低, 那么 Bob 可以通过观察 $H(q_i)$ 来猜测 (或检查)s 的某些比特. 那么, 如何保证 Bob 在协议中使用的 $R_i \in \{0^n, 1^n\}$

呢? Keller, Orsini, Scholl 发表于 2015 年美密会的一致性检查 (Consistency check) 技术可以做到这一点. 那么 KOS 协议在原来的基础上做了什么改进呢? 简单来说就是加入了一致性检查.

优化方案

如下图所示, 由于 IKNP 等式 $q_i = t_i \oplus R_i \wedge s$ 对矩阵的所有行 i 都成立且所有行的 s 都相同, 因此这些行数集合组成的子集 C 所对应的式子求异或之和后该等式仍然是成立的, 即有以下等式

$$\left(\bigoplus_{i\in C} q_i\right) = \left(\bigoplus_{i\in C} t_i\right) \oplus \left(\bigoplus_{i\in C} R_i\right) \wedge s \tag{1}$$

为此, Alice 可以发起挑战, 选取随机行数组成的集合 C 发给 Bob. Bob 计算 $t^* = \bigoplus_{i\in C} t_i$, $R^* = \bigoplus_{i\in C} R_i$, 并发送 t^*, R^* 给 Alice 作为应答. 由于 Alice 有 s 和 q_i, 因此她可以计算 $q^* = \bigoplus_{i\in C} q_i$, 并验证等式 (1) 是否成立.

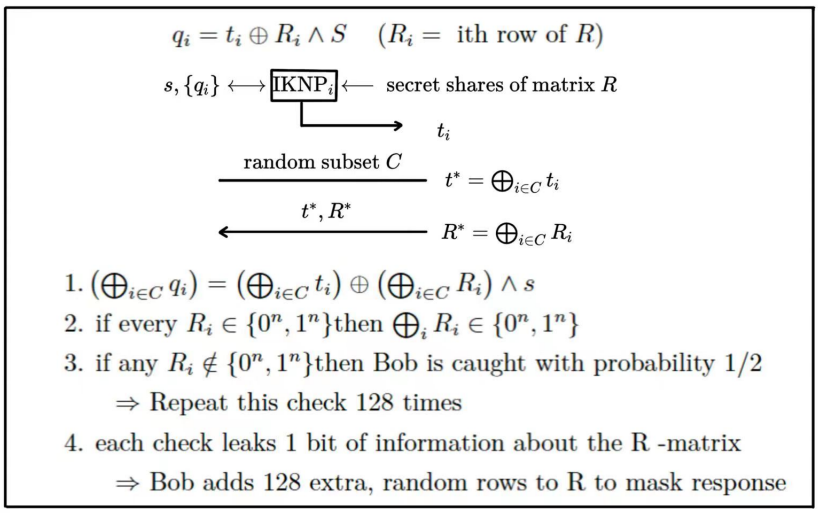

Figure 4: KOS 协议

KOS 协议的一致性检查的主要思想在于如果 $R_i \in \{0^n, 1^n\}$, 那么对于行的任意组合来说其异或和的结果必然也属于 $\{0^n, 1^n\}$ 这个集合; 如果存在某个 R_i 不属于这个集合, 那么 Bob 将有 1/2 的概率通过一致性检查从而作弊成功, 此外, Bob 也可以通过猜测 s 使其发送的信息 t^*, R^* 通过一致性检查, 为此只需将一致性检查重复 $\lambda = 128$ 次, 如果 Bob 能通过所有的一致性检查, 则 Alice 可以认为 Bob 输入的 R_i 确实是属于 $\{0^n, 1^n\}$ 这个集合的. 但是, 该一致性检查技术的每次检查会泄露矩阵 R 的一比特信息, 为此 Bob 只需添加 $\lambda = 128$ 个额外的随机行到 R 中来盲化应答.

4 应用场景

我们在第一节进行引入时就已经涉及到了 MPC 中使用到 OT 的始祖级协议: 姚氏混淆电路, 在这个协议的构造过程中, 混淆方需要将解密方的真实值对应的混淆值发送给解密方, 但一是混淆方不能知道这个真实值, 二是解密方不能获得另外一个混淆值, 这恰好就是非常典型的两方不经意传输的情况, 接下来我们来看另外一个协议中不经意传输是怎么起作用的:

4.1 BMR 协议

Beaver, Micali, and Rogaway (BMR) 协议是安全多方计算的另一个重要协议，它将 Yao 的混淆电路和 OT 等技术扩展到了多方计算的情况。在 BMR 协议中，OT 起到了关键的作用。

BMR 协议的基本思想是每个参与者都为要计算的函数构建一个混淆电路，并且这些电路应该在语义上是等价的。然后，每个参与者将他们的电路和输入共享给其他所有的参与者。为了保护每个参与者的输入，这个过程需要使用 OT。

BMR 协议大致可以分为以下步骤：

生成和混淆电路：每个参与者都为要计算的函数生成一个二进制的电路，并且混淆这个电路。混淆的电路应该在语义上等价，也就是说它们应该对同样的输入给出同样的输出。

共享混淆电路：每个参与者将他们的混淆电路共享给其他所有的参与者。这样每个参与者都有了所有的混淆电路。

输入选择和共享：每个参与者选择他们的输入，并且使用 OT 将这些输入安全地共享给其他所有的参与者。这个过程保证了每个参与者的输入对其他参与者保持私密。

计算和验证：每个参与者使用他们接收到的所有电路和输入，计算出结果。由于所有的电路在语义上是等价的，所以如果没有出错或者恶意行为，所有的参与者应该计算出同样的结果。

这就是 BMR 协议的基本过程，其中 OT 的作用主要是在输入选择和共享的阶段，保证了每个参与者的输入对其他参与者保持私密。接下来我们对协议交互流程进行一个展示，由于本文重点不在于 MPC 因此此处不对 BMR 协议作原理解释。

协议流程简述

首先根据需要实现的目标函数生成布尔电路 $\mathcal{C}$，并向所有参与者公开。将电路 $\mathcal{C}$ 中的各条导线分别标记为 $w_1, w_2, \ldots, w_{\mathrm{t}}$。协议参与者为 $P_1, P_2, \ldots, P_n$，其输入分别为 $x_1, x_2, \ldots, x_n \in \{0,1\}$。

对于电路 $\mathcal{C}$ 中的每一条导线 w_i，参与者 P_j 随机产生导线 w_i 的真值 0，1 对应的随机值，分别记为 $w^0{}_{\mathrm{i,j}}, w^1{}_{\mathrm{i,j}}, w^{\mathrm{b}}{}_{\mathrm{i,j}} = \left(\mathrm{k}^{\mathrm{b}}{}_{\mathrm{i,j}}, \mathrm{p}^{\mathrm{b}}{}_{\mathrm{i,j}}\right) \in \{0,1\}^{1+1},\ \mathrm{b} \in \{0,1\}$

$p^b_{i,j}$ 满足$p^b_{i,j} = 1 - p^{1-b}_{i,j}$ ，即 $k^b_{i,j}$ 为 l 比特的随机比特串，$p^b_{i,j}$ 为 1 比特的随机比特。再产生一个随机翻转比特 $f_{i,j} \in 0,1$。对于每一条导线，参与者 P_i 计算 $\mathrm{S}_{\mathrm{i,j}} = \left(\mathrm{F}\left(w^0{}_{\mathrm{i,j}}\right), \mathrm{F}\left(w^1{}_{\mathrm{i,j}}\right), \mathrm{p}^0{}_{\mathrm{i,j}}, \mathrm{f}_{\mathrm{i,j}}\right)$。因为 $p^b_{i,j} = 1 - p^{1-b}_{i,j}$，因此只需要发送 $p^0_{i,j}$ 或者 $p^1_{i,j}$ 就可。函数 $F()$ 可看作是哈希函数，其将 l 位的输入扩展为 $n \times (l+1)$ 位，n 是参与者数量。

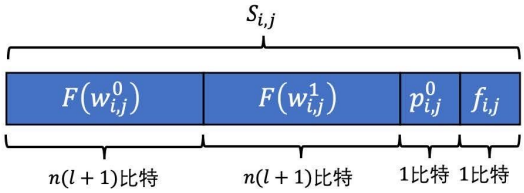

Figure 5: 预计算部分

共同计算部分

对于布尔电路 $\mathcal{Q}$ 的每一个门 G_i ，所有参与者共同参与一个计算乱码表的多方安全计算协议。该协议的输入是每个参与者之前计算的 $S_{i,j}$ 和参与者的输入 $x_1, x_2, \ldots, x_n$ 。该协议的目标为：假设电路 $\mathcal{C}$ 中门 G_i 的输入导线为 w_a, w_b，输出导线为 w_c。门 G_i 实现的函数为 $w_c = g\left(w_a, w_b\right)$。 。

计算指针比特：

$p^0_a = \oplus_{j=1,\ldots,n} p^0_{a,j}, p^0_b = \oplus_{j=1,\ldots,n} p^0_{b,j}, p^0_c = \bigoplus_{j=1,\ldots,n} p^0_{c,j}$

并设置：

$p_a^1 = 1 - p_a^0, p_b^1 = 1 - p_b^0, p_c^1 = 1 - p_c^0$

同理，对所有参加者提交的翻转比特求异或

$f_a = \bigoplus_{j=1,\ldots,n} f_{a,j}, f_b = \bigoplus_{j=1,\ldots,n} f_{b,j}, f_c = \bigoplus_{j=1,\ldots,n} f_{c,j}$

得到翻转比特 f_a, f_b, f_c，翻转比特用于掩盖参与者 P_i 的输入，使得 P_i 无法得知其对每条导线加密值的具体贡献。接着创建门 $\mathcal{G}_i$ 的乱码表，让 v_a, v_b 分别是导线 w_a, w_b 的输入，对于 $\mathcal{G}_i$ 输入 v_a,$v_b \in \{0,1\}$，一共只有四种可能的组合。

让 v_c 是导线 w_c 的真实输出值，e_{v_a,v_b} 是导线 w_c 的加密值。设定

$e_{v_a,v_b} = w_c^{v_c \oplus f_c} \bigoplus_{j=1,\ldots,n} \left(F\left(i, w_{a,j}^{v_a \oplus f_a}\right) \oplus F\left(i, w_{b,j}^{v_b \oplus f_b}\right)\right)$

即 e_{v_a,v_b} 共有四个可能的值，如下表所示：

v_a	v_b	e_{v_a,v_b}
0	0	$e_{0,1} = w_c^{v_c \oplus f_c} \bigoplus_{j=1,\ldots,n} \left(F\left(i, w_{a,j}^{0 \oplus f_a}\right) \oplus F\left(i, w_{b,j}^{0 \oplus f_b}\right)\right)$
0	1	$e_{0,1} = w_c^{v_c \oplus f_c} \bigoplus_{j=1,\ldots,n} \left(F\left(i, w_{a,j}^{0 \oplus f_a}\right) \oplus F\left(i, w_{b,j}^{1 \oplus f_b}\right)\right)$
1	0	$e_{1,0} = w_c^{v_c \oplus f_c} \bigoplus_{j=1,\ldots,n} \left(F\left(i, w_{a,j}^{1 \oplus f_a}\right) \oplus F\left(i, w_{b,j}^{0 \oplus f_b}\right)\right)$
1	1	$e_{1,1} = w_c^{v_c \oplus f_c} \bigoplus_{j=1,\ldots,n} \left(F\left(i, w_{a,j}^{1 \oplus f_a}\right) \oplus F\left(i, w_{b,j}^{1 \oplus f_b}\right)\right)$

Figure 6: BMR 协议

其中 $\mathrm{w_c^b} = \mathrm{w_{c,1}^b} \| \cdots \| \mathrm{w_{c,n}^b} \| \mathrm{p_c^b}, \mathrm{b} \in \{0,1\}$, 即将各个 $\mathrm{w_{c,i}^b}$ 和 $\mathrm{p_c^b}$, 进行级联。之后对乱码表中 $e_{va,vb}$ 的条目进行重新排序，并将条目 $e_{va,vb}$ 放在 $(\mathrm{p_a^{va}}, \mathrm{p_b^v})$ 的位置上。之后各参与方向参与者 P_1 输出计算得到的乱码表，向 P_1 发送各个参与方的输入 $x_1, x_2, \ldots, x_n$ 所对应的电路 $\mathcal{C}$ 的加密导线值。据此 P_1 可以根据所收到的信息恢复出秘密计算结果。

当然，在当下安全多方计算中广泛使用的更现代的协议 SPDZ 中，也同样使用了不经意传输（OT）作为一种关键构件。SPDZ 协议由 Damgård 等人在 2012 年提出，该协议可以支持任意数量的参与者进行安全多方计算，并且可以抵抗主动攻击。在这个协议中，OT 主要用于在预处理阶段实现共享随机乘积的生成。

这个协议主要由两个阶段组成：

预处理阶段：这一阶段在协议开始之前运行，生成随机共享以及随机共享的乘积，为在线阶段提供预计算的材料。在这个阶段中，每对参与者之间都需要运行 OT 协议，生成随机共享的乘积。

在线阶段：在这个阶段中，参与者使用预处理阶段生成的预计算材料，通过线性操作和乘法操作执行计算任务。由于预处理阶段已经生成了随机共享的乘积，所以在在线阶段的乘法操作可以直接使用这些乘积，大大提高了效率。

SPDZ 协议通过预处理阶段和在线阶段的分离，以及在预处理阶段使用 OT 生成随机共享的乘积，实现了在多方计算中高效率和高安全性的平衡。以上种种均说明 OT 在 MPC 协议构建中的重要地位。

5 近期发展方向和关于 OT 的一些思考

通过前述内容，我们已经对不经意传输有一个初步的了解。这个部分我首先会结合最近在看的几篇文章简单叙述一下几个业内比较有趣的发展趋势。接着，我们对不经意传输到底为什么工作，二是不经意传输究竟为什么那么通用这两个问题做一个简单的探讨。

5.1 几个有趣的发展趋势

5.1.1 带有隐藏访问控制策略的不经意传输

这个想法来源于文章"Oblivious Transfer with Hidden Access Control Policies." Miao, Furong, et al. Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. 2019. 论文探讨了是在 OT 协议中实现隐藏的访问控制策略的方法。这意味着发送方可以根据某些隐藏的规则来控制接收方可以接收哪些信息，而不必直接告诉接收方这些规则。这种做法可能会进一步提高 OT 的隐私保护能力。这篇文章反映了以下关于 OT 发展的有趣趋势：

复杂度增加：OT 的应用正在变得越来越复杂。在传统的 OT 中，发送方只需要决定要发送哪些信息，接收方选择接收哪些信息，而在这篇论文中可能提出的隐藏访问控制策略下，发送方需要制定并执行复杂的规则来决定接收方可以接收哪些信息。这表明了 OT 在处理更复杂的应用场景方面的潜力。

隐私保护的增强：OT 的核心目标是保护隐私，而隐藏的访问控制策略可能会进一步提高这种隐私保护能力。这可能反映了隐私保护在当前 OT 研究中的重要性，也可能指示了未来的研究方向，即如何更有效地保护隐私。

个性化的需求：随着 OT 的广泛应用，用户可能有更多的个性化需求，例如，他们可能希望能够更细致地控制信息的接收。隐藏的访问控制策略可能是满足这种需求的一种方法，这可能反映了 OT 在满足更个性化需求方面的发展趋势。

5.1.2 适应特异化 MPC 框架的 OT 协议

这个想法在"Fast Actively Secure Five-Party Computation with Security Beyond Abort" 这篇文章中得到了突出体现，其专注于解决在五方计算中实现快速、主动安全的协议，同时其安全性超越了简单的发现恶意行为后立刻中止。在多方计算中，多个参与者可以合作计算一个函数，而每个参与者只知道自己的输入，不知道其他参与者的输入。这种计算模型为处理和分析敏感数据提供了可能，但它也带来了一系列的安全挑战。其中一个主要挑战是如何在参与者可能作恶的情况下保证计算的正确性和隐私性。

这篇文章提出了一种新的五方计算协议，这个协议可以在参与者可能作恶的情况下保证计算的正确性和隐私性，同时还有良好的效率性能。文章使用了一个特制的不经意传输协议作为基本的构建模块，实现了非常良好的传输性能。这篇文章反映了 OT 发展的一些主要趋势：

OT 的应用范围正在扩大: 原来，OT 主要被用于两方计算，现在，研究者开始探索如何将 OT 用于多方计算。这显示了 OT 的潜力和灵活性。

OT 的效率和安全性都在得到改进: 为了满足实际应用的需要，研究者正在开发新的 OT 协议，这些协议既要具有高效的性能，也要具有强大的安全性。

OT 的安全模型正在变得更加复杂和细致: 早期的 OT 协议主要关注抵抗被动攻击的能力，现在，研究者开始考虑如何在参与者可能作恶的情况下保证 OT 协议的安全性。

5.2 为什么能工作？

首先我们我们来探究一下不经意传输为什么是能够工作的。还是回归 OT 的要求:

1. Alice 不知道 Bob 的选择.

2. Bob（完全）不知道另一条消息.

为了达成这个目的，收方和发方各需要进行一次置乱以保证上面的这两个性质得到满足，结合前文我们所讲的各类协议可以感性的认识到这些协议的思路基本都是双方对同一个公用消息进行加密，但是在此基础上收方的加密保留了一定的 “后门”，也就是可以在有限范围内获得到发方加密的消息的信息，所以和追求严格加密安全性的大部分密码学协议相比，个人感觉 OT 协议所追求的是一个有限的后门函数，颇有废物利用的感觉. 当然，这种部分的隐私泄露也导致构造一个真正意义上非常安全的 OT 协议非常困难，回到上面提到的两个要求:

前者其实没有太大的问题，就是要求 $Enc_{pk_1}(K)$ 和 $Enc_{pk_2}(K)$ 不可分辨。后者令人困惑，明明两条消息都发出去了，Bob 会不知道另一条消息吗，或者说 Bob 会完全不知道另一条消息吗? 其实所有的 OT 实现这里都作了个弊，我们无法构造出一个 Bob 完全不知道另一条消息的方法。我们说的完全是什么呢? 如果发送方对另一条消息就发一个随机数，那么接收方除了暴力穷举一遍没有任何办法，蒙对消息的可能性就是 $\frac{1}{2^k}$。但 OT 不同，比如我们恶意安全的 OT 方案，Bob 不是完全不知道另一条消息，而是很困难。再看一遍我们的协议，Bob 是知道另一个 pk 的，他只要能够根据 pk 得到 sk 就可以得到 G，从而接触另一条信息。而从 pk 到 sk 就是一个困难问题，而非不可能问题，这正是公钥密码学的基础。而这些困难问题的背后，都隐藏着一个单向陷门函数.

5.3 不经意传输通用性

我们已经知道了不经意传输（OT）几乎是所有多方安全计算协议背后的基础，混淆电路可以以 OT 来构造，乘法秘密分享，隐私求交（PSI），隐私信息查询（即俗称匿踪查询，PIR）这些背后都有 OT 的身影。那么是什么导致了 OT 的通用性?

我们先对通用性做个说明。通用性我们可以举几个例子，比如 NAND 或 NOR 对于布尔代数是自足的（functional complete), Lambda Calculus 只需要 variable, abstraction, application，欧式几何只需要五个公设等。所以如果 OT 通过混淆电路可以做通用的逻辑电路，似乎已经证明了它的通用性.

另一个问题来源于隐私计算的实用性要求。对于 PSI，或者联合统计等应用，纯粹以昂贵的混淆电路去做是不现实的，而采用 OT，尤其是当利用 OT 扩展时，OT 的成本大大降低，直接去构造 OPRF，或者三元组，这就将隐私计算的实用性向前推进了一大步。时至今日，OT+YAO's GC 的组合仍然是效率和成本非常优越的方案，这也就是其通用性的由来.

5.4 后量子密码语境下的 OT

如同我们在 5.2 节中所提到的，接收方恢复另外一条信息内容的概率永远是极低而非不可能，其原因就在于两条消息的绑定传输本身就已经泄露了一定的信息量，因此很多 OT 协议的困难性最终都会归结于我们目前广泛使用的数学困难问题，但在量子计算机的威胁下这一模型似乎并不安全。关于量子计算对现有密码学系统的威胁，大多数都源自量子计算能力的两个主要特性：量子纠缠和量子叠加。这使得量子计算机可以在某些特定类型的问题上超越经典计算机，包括一些与密码学直接相关的问题。

对于不经意传输，最直接的威胁来自量子计算机对于数因子分解和离散对数问题的高效解决能力。这两个问题在经典计算上被认为是困难的，因此被用作许多密码学协议（包括某些 OT 协议）的安全基础。然而，量子计算机可以通过 Shor 的算法在多项式时间内解决这两个问题，这就对依赖这些问题困难性的 OT 协议构成威胁。

针对这种威胁，研究者已经开始发展新的量子抗性密码学协议，这些协议的安全性不依赖于这两个问题的困难性，而是依赖于其他在量子计算模型下仍然被认为是困难的问题。在后量子密码学中，已经有一些 OT 协议被提出，这些协议可以抵抗量子攻击。这些协议的安全性通常基于一些在量子计算模型下被认为是困难的数学问题，如 LWE 或编码理论中的某些问题。不过，虽然这些新的 OT 协议可以提供量子安全性，但它们通常会比经典的 OT 协议更加复杂，且效率更低。因此，对于现有的系统，是否需要升级到这些新的协议需要进行综合考虑，包括当前量子威胁的具体情况、系统的安全需求和性能需求等因素。

6 亟待解决的问题和研发方向

无论是侧重于效率优化的半诚实敌手模型, 还是效率较低的恶意敌手模型, 协议本身的改进加上工程实现技术的成熟使得近年来 OT 协议、OT 扩展协议在安全性和实用性上都取得了显著的研究进展, 推动了 SMPC 技术在不同应用场景下的落地, 比如隐私保护集合交集运算. 相信在不久的将来我们有望看到可靠高效的 SMPC 技术在日常生活中得到普及应用, 为数据安全与隐私保护领域作出更多贡献.

在当前的不经意传输（Oblivious Transfer，OT）协议及其扩展研究的基础上，我们发现还有很多潜在的研究方向值得探索：

1. 设计满足统一模型（Universal Composability，UC）安全性的高效 OT 扩展协议和 Correlated OT（COT）扩展协议是非常重要的。统一模型是一种强大的安全框架，可以保证在并行执行多个协议的情况下维持其安全性。设计出满足这种强安全性质的 OT 协议是一个关键挑战。此外，这些新设计的协议如何在多方安全计算协议中应用和测试，也是一个值得探索的问题。

2.Silent OT 和 Ferret OT 扩展协议通过解决了通信开销的瓶颈，为 OT 的实际应用开启了新的可能。研究如何将这两种 OT 扩展协议应用到不同的安全多方计算协议（Secure Multi-Party Computation，SMPC）以及机器学习隐私保护方案中，进行开销对比和可用性分析，将会对实际应用有实质性的影响。

3. 基于 Pseudorandom Code Generator (PCG) 的 OT 扩展协议框架，可以被用来构造高效的 1-out-of-n OT 扩展协议。1-out-of-n OT 是一个基本的 OT 变体，其中接收者可以选择一个发送者的 n 个输入，而发送者并不知道接收者选择的是哪一个输入。高效的 1-out-of-n OT 扩展协议有着广泛的应用前景。

4. 结合分布式的思想实现 OT 扩展协议，这是一个很有潜力的研究方向。在分布式环境中实现 OT，既可以提高系统的容错性，又可以在一定程度上提高系统的性能。

5.rate-1 OT 是 OT 的一种变体，其中每一次 OT 操作可以传输一个完整的消息位，而不是仅仅传输一个消息位的一部分。如何优化 rate-1 OT 的计算和通信效率、如何将 rate-1 OT 的思想与 OT 扩展协议相结合，以及如何研究其他的 rate-1 OT 变体，都是值得探索的问题。

6. 使用最新可用的工程优化技术进行实现对比，可以有效地提升 OT 协议的实际性能。在这方面，既可以使用已有的工程优化技术，也可以探索更多新的、易于普及的工程优化思路。

7 总结

在本文中，我们对不经意传输的概念和发展进行了回顾，并在叙述过程中介绍了该领域意义重大的研究成果。同时，我们对于不经意传输的安全性、仍然存在的问题和发展趋势进行了讨论。总体而言，OT 提升的主要技术点在于更高的传输效率、更加专用化以及更优的安全性。现有的协议在以上方面均还有一定欠缺，以上种种均对 OT 组件的性能有所限制。因此，如何优化这些方案是一个值得长久进行研究的问题，我们期待在未来看到在该领域有更多精妙的设计与喜人的成果。

最后感谢您阅读至此，希望本文对您有所启发。

References

[1] Rabin M O . How to Exchange Secrets by Oblivious Transfer[J]. Technical Memo TR-81, 1981.

[2] Even S . A randomized protocol for signing contracts[J]. ACM SIGACT News, 1983.

[3] Brassard G , C Crépeau, Robert J M . All-or Nothing Disclosure of Secrets. Advances in Cryptology —CRYPTO' 86, 1986.

[4] Donald Beaver. Efficient Multiparty Protocols Using Circuit Randomization. CRYPTO 1991.

[5] Yuval Ishai, Joe Kilian, Kobbi Nissim, Erez Petrank. Extending Oblivious Transfers Efficiently. CRYPTO 2003.

[6] Marcel Keller, Emmanuela Orsini, Peter Scholl. Actively Secure OT Extension with Optimal Overhead. CRYPTO 2015.

[7] Vladimir Kolesnikov, Ranjit Kumaresan, Mike Rosulek, Ni Trieu. Efficient Batched Oblivious PRF with Applications to Private Set Intersection. CCS'16.

[8] Vladimir Kolesnikov, Ranjit Kumaresan. Improved OT Extension for Transferring Short Secrets. CRYPTO 2013.

[9] Beaver, D., Micali, S., & Rogaway, P. (1990). The Round Complexity of Secure Protocols. In Proceedings of the twenty-second annual ACM symposium on Theory of computing (pp. 503-513).

[10] Alhassan, Ashraf, et al. Fast Actively Secure Five-Party Computation with Security Beyond Abort. IACR Cryptol. ePrint Arch. 2020: 758. 2020.

[11] Miao, Furong, et al. Oblivious Transfer with Hidden Access Control Policies. Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. 2019.

[12] David Evans, Vladimir Kolesnikov, Mike Rosulek. A Pragmatic Introduction to Secure Multi-Party Computation.