Oblivious Transfer 总结

不经意传输（oblivoius transfer）是一个密码学协议，目前被广泛的应用于安全多方计算（SMPC，Secure Multi-Party Computation）。它由 Rabin[1] 在 1981 年提出。

Rabin 1981 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：

> 方案假设两方的秘密都是单比特!!
>
> $(1)Alice$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$c \equiv x^2 mod\ n_A$，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$x_1^2 \equiv c \mod (n_A)$，发送$x_1$给$Bob$。
>
> $(4)Bob$计算$gcd(x-x_1, n_A) = d$，此时可以证明的是$p(d == q\ or\ d == q) = \frac{1}{2}$
>
> $(5)Bob$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$\varepsilon_B = v_B \bigoplus S_B$ 然后将 ε_B发送给$Alice$

这是 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$\frac{1}{2}$。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。

可以得出 Rabin 提出的方案两方都无法获得对方的秘密的概率和两方同时获得对方秘密的概率都是$\frac{1}{4}$。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，还不具有应用意义。所以有了 1985 年 Even[2] 等人在此[1]基础上提出的新的 1-out-2 OT 协议[2]。

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-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$\frac{1}{2}$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。如下图：

具体过程为：

> (1)$Alice$在公钥密码中随机选择一组密钥(E_x, D_x);
>
> 同时随机选择加密算法明文空间的两个明文m_0, m_1;
>
> 最后将E_x, m_0和m_1发送给Bob.
>
> (2)Bob随机选择$r \in \{0, 1\}$;
>
> 然后在与(1)同样的明文空间选择k;

> 计算$q = E_x(k) \boxplus m_r$，并且向$Alice$发送q;
>
> (3)$Alice$计算$k'_i = D_x(q \boxminus m_i)$，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$Bob$.
>
> (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；
>
> s, r相同选第一项，不同选第二项，与自己k进行$\boxplus$运算得到秘密$M_{s \oplus r}$.
>
> $\boxplus$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，OT_2^1 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。OT_2^1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。

1986 1-out-of-n OT

之后在1986年，Brassard[3] 等人继续将OT协议改进到了 1-out-n OT 版本。与上述[2]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 .

初始协议为：n条秘密为：$x_1, x_2, \ldots, x_n$,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$z_{i,j} = x_{i,j}^2 y^{b_{i,j}} mod\ m$ （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$q = z_{i,j} r^2 y^{\alpha}\ mod\ m$，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，由此可推断出秘密 i 。

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，改进上述算法如下：

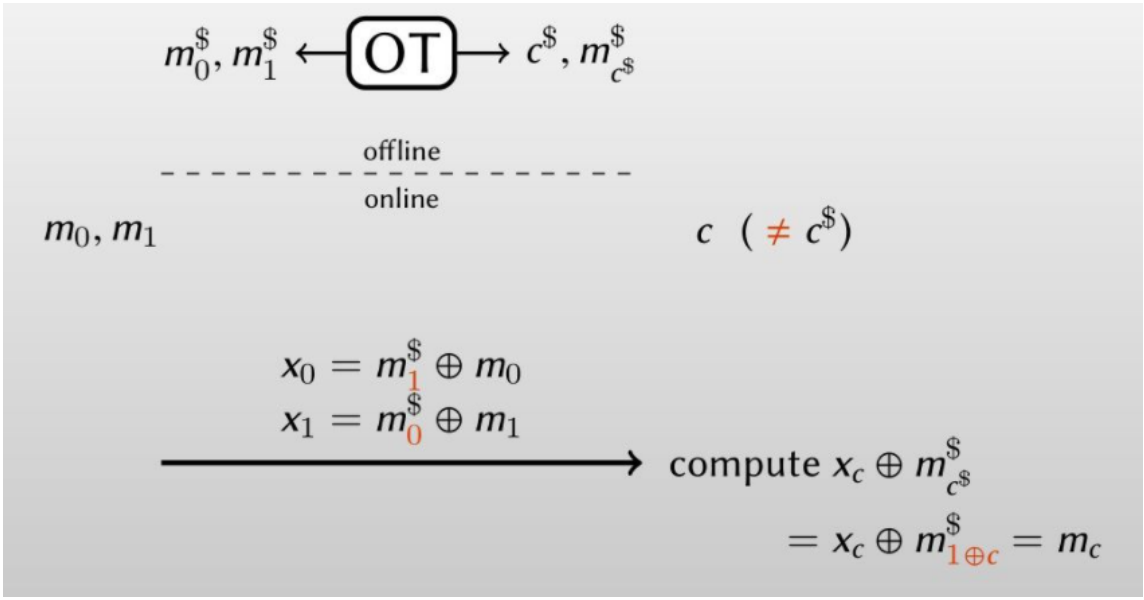
> (1) Bob 随机选择扰动函数 $\varepsilon()$，随机整数$r_{k,j}$和随机比特位 $\alpha_{k,j} \in \{0,1\}$，使得 $k = \varepsilon(i)$，其中 i 表示 Bob 想要获得第 i 条秘密。并计算$q_{k,j} = z_{i,j} r_{k,j} y^{\alpha_{k,j}}\ mod\ m$ 将 t 个$q_{k,j}$ 传送给 Alice，同时需要向 Alice 证明所有的 $q_{k,j}$可用性（我理解以此来保证请求的是同一条消息的不同比特位，也就是解决缺点 1和2）。
>
> (2) Bob 传送 k 给 Alice（$k = \varepsilon(i)$).
>
> (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根据二次特征值来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OT_n^1协议。至此最基本的OT协议包括2取1，n 取 1 都已经有了。接下来就是关于OT扩展的内容了。

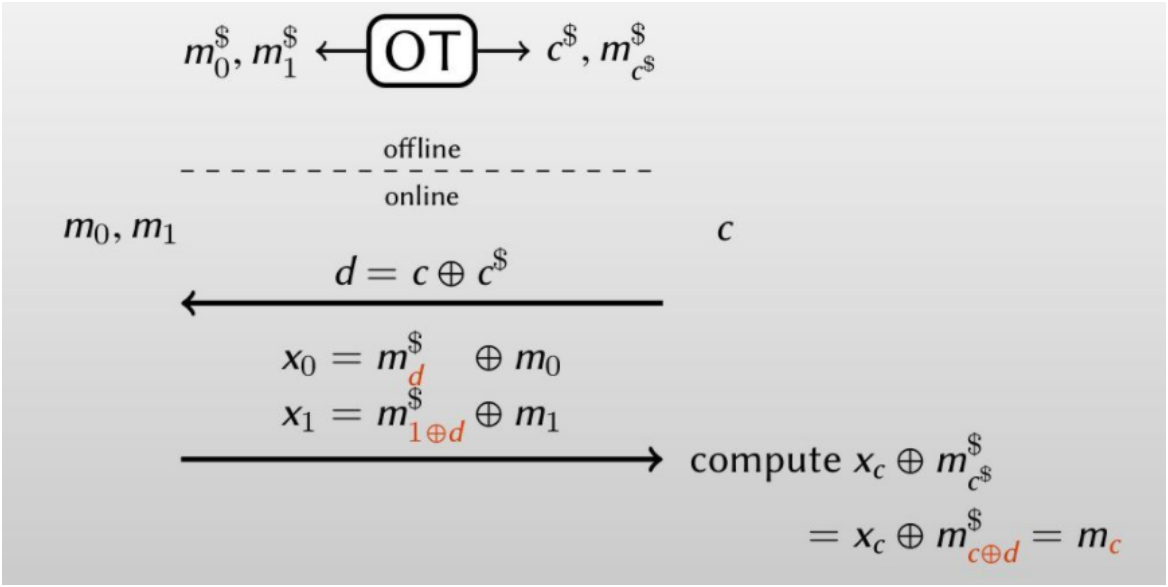
Beaver 96

但是在1988年，Impagliazzo和Rudich[4] 就已经证明了不能使用黑盒的构造方法从一个单项函数来实现OT，也就是说，虽然OT协议的设计已经很成熟，但是应用中的运算消耗仍然很高。而公钥密钥的出现有助于解决现有的OT协议执行次数过高的问题，不过因为公钥密钥需要低效率的幂指数运算，不利于直接加密大数据，因此将公钥的私钥两种方法结合，对现有OT进行扩展，提高OT协议的效率。1996年Beaver[5]等人就使用这种方法，提出了一种OT扩展技巧，通过伪随机装置来对等大量OT执行的效果。伪随机依靠一个单向函数（一个黑箱）来了实现。

下图是一个基本的 OT_2^1, 设预计算阶段的 R-OT 协议使发送方 Alice 获得两个随机信息m_0^r, m_1^r, 接收方Bob获得$c^r, m_{c^r}^r$, 这里的上标 r 与图中的 δ 含义相同, 设在线阶段Alice的需要发送的信息为m_0, m_1, Bob 的选择比特为 c . Beaver 去随机化的主要思想是 Alice 使用R-OT的两个随机信息 m_0^r, m_1^r作为一次一密(OTP)来加密需要他发送的信息 m_0, m_1并将盲化结果 (x_0, x_1)发给 Bob. 如果 $c = c^r$, Alice发送的两个信息是 $x_0 = m_0^r \bigoplus m_0, x_1 = m_1^r \bigoplus m_1$, Bob本地计算$x_c \bigoplus m_{c^r}^r = x_c \bigoplus m_c^r = m_c$即可得到选择比特 c 对应的比特秘密 m_c ,但是无法获得 m_{1-c} 。但是如果 $c \neq c^r$ ， 按照上述步骤我们会发现 Bob 将无法获得正确的信息，除非 Alice 交换 m_0^r, m_1^r 来加密 m_0, m_1 。

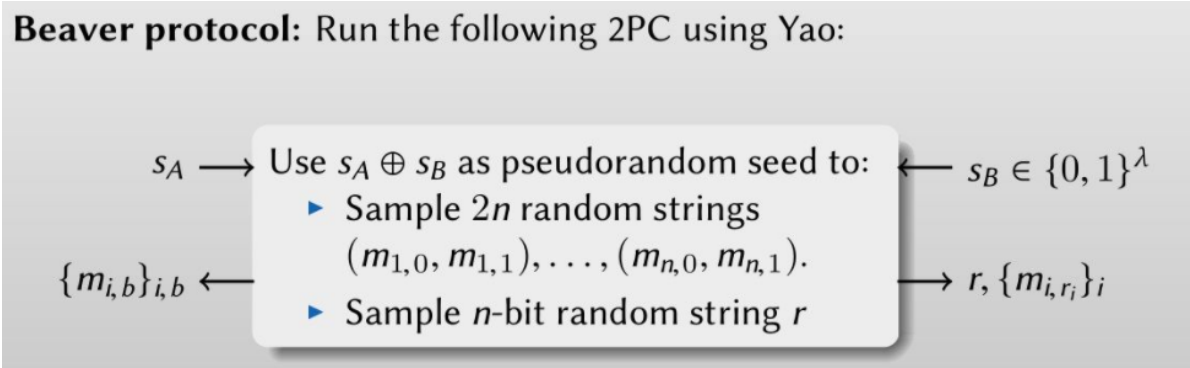


为解决上述 $c \neq c^r$ 的情况，易得的解决方法是让 Bob 告知 Alice 是否有 $c = c^r$ ，具体如下图所示，其中$c = c^r$是否等于0，表示为$d = c \bigoplus c^r$ 表示，



上述过程就是一个在基本的 OT_2^1 上使用随机数来实现的过程，即 R-OT 。但是其每执行以此 OT 都需要进行里现阶段的随机数生成，这个开销非常昂贵。反之其在线阶段只需要异或运算，反而比较高效。并且以此离线阶段的R-OT以此只能产生一个OT实例，这也给其应用带来困难。而Beaver等人在1996年，使用了混合加密的方式对R-OT进行了改进，使得其离线阶段效率得到了提高，并且例证了OT扩展方案的可行性。

要讲清Beaver等人的 OT 扩展，首先了解一下混合加密。公钥密钥的出现有助于解决现有的OT协议执行次数过高的问题，不过因为公钥密钥需要低效率的幂指数运算，不利于直接加密大数据，因此将公钥的私钥两种方法结合，，使用昂贵的公钥加密短密钥，然后使用相对便宜的私钥来加密长信息。以此实现对现有OT进行扩展，提高OT协议的效率。应用如下：



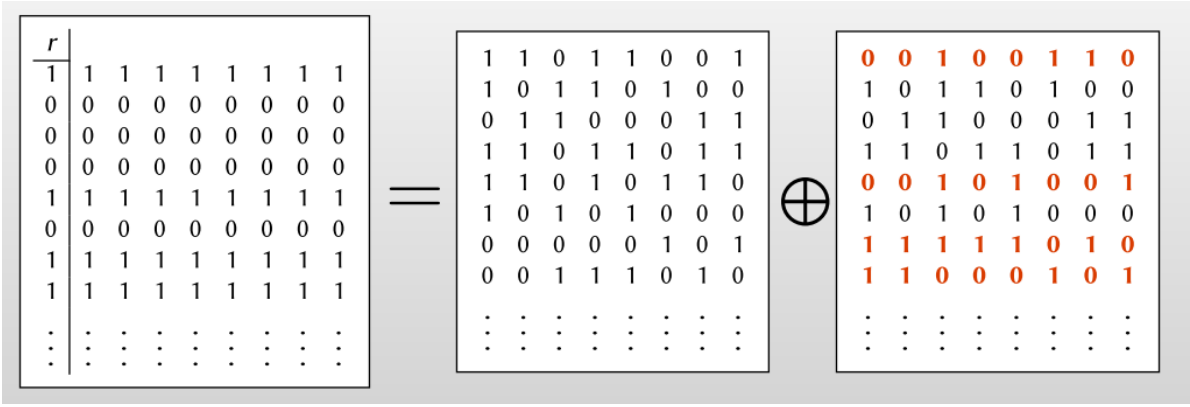
如上图所示，使用姚氏混淆电路来完成真个过程，其中左侧为 Alice 输入 S_A 并得到所有的消息对，即就是 n 个R-OT离线阶段的 (m_0, m_1)，右侧为 Bob 输入 S_B 并得到比特串 r 以及对应的消息 (r, m_r)，即就是R-OT离线阶段的$(c^\$, m_{c^\$}^\$)$。其中输入的$S_A, S_B$长度为$\lambda$，输出的结果个数为 n，且 $n \gg \lambda$。也就是说在此扩展中只需要 λ 个R-OT，可以得到 n 个 OT 实例。

然而由于姚氏混淆电路的原因导致其不够实用，但是他例证的说明了 OT-extension 是可行的。也为后面 OT 的发展做出重要贡献。

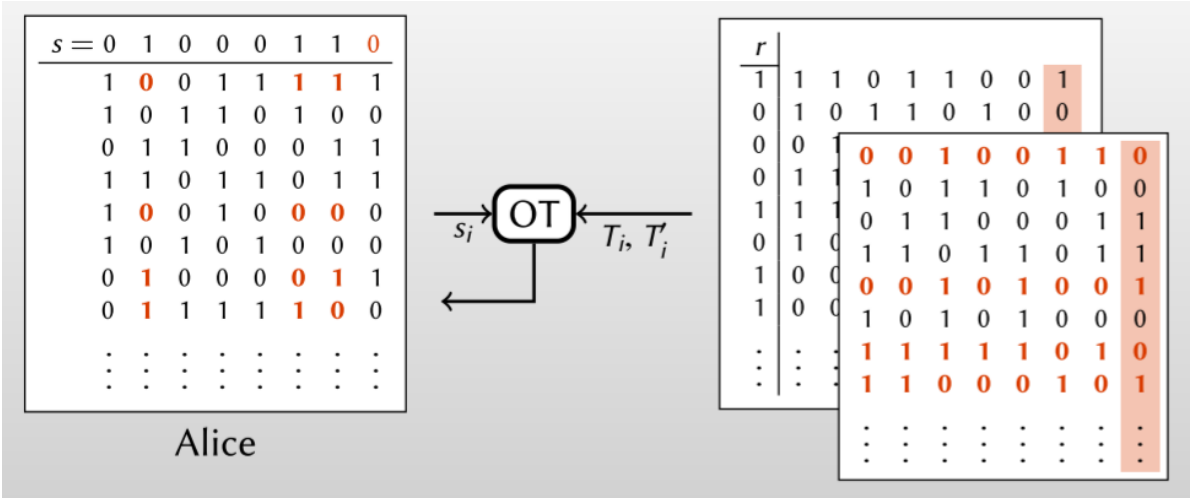
IKNP02

在Beaver等人之后，IKNP03[6] 协议则逐步让 OT 协议走上实用的道路。IKNP03协议和Beaver等人解决的问题实际上是相同的，都是为了让 Alice 获得消息对 (m_0, m_1) 和让 Bob 后的对应的 $(r,\ m_r)$ 以便在线阶段 OT 的使用。与不太实用的Beaver96使用 Yao's GC 不同的是，IKNP03 使用 OT 来实现，具体如下：

首先，Bob 随机构造长度为 n 比特串 r ,然后将其看为一个 $n \times 1$ 的列向量对每个比特位进行按行扩展为 λ (重复编码扩展)，完成后进行秘密分享，如图：



接着 Alice 也随机选取长度为 n 的比特串 s，然后双方共同执行base-OT 其中 Bob 为发送方，将两个秘密份额矩阵的第 i 列作为输入的两个消息，Alice 作为接收方，以 s_i 为输入来选择两个秘密份额矩阵第 i 列的某一个来构成自己的矩阵 Q 的第 i 列，结果如下:

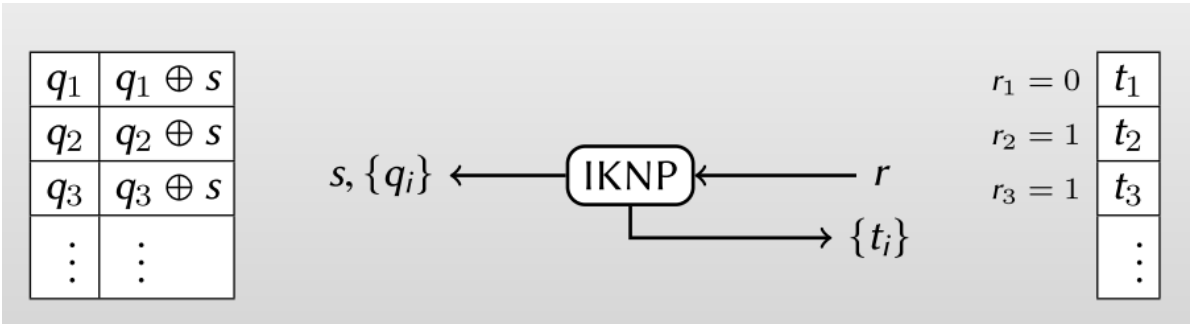


观察可以发现，当 $r_i = 0$ 时, Alice所得矩阵 Q 的行向量 q_i与 Bob 的份额矩阵该位置对应的行向量 t_i 相同；当 $r_i = 1$ 时, Alice所得矩阵 Q 的行向量 q_i等于 Bob 的份额矩阵该位置对应的行向量 t_i 与 s 的异或，即：

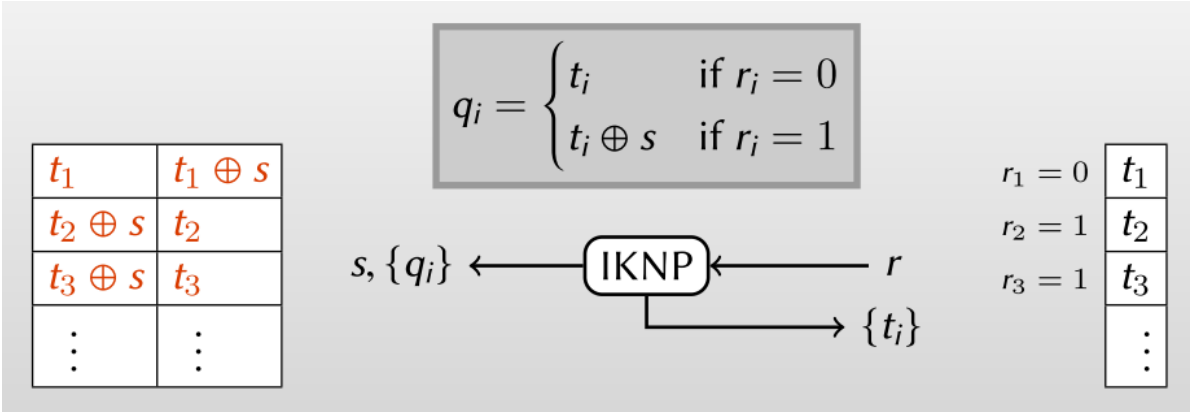
$$q_i = \begin{cases} t_i & \text{if } r_i = 0 \\ t_i \bigoplus s_i & \text{if } r_i = 1 \end{cases}$$

也就是说上述过程可以抽象为： $q_i = t_i \bigoplus (r_i \wedge s_i)$

上述过程后 Alice 获得了矩阵 Q ，加上自己拥有的选择串 s，则 Alice 就有了消息对 $(q_i, q_i \bigoplus s)$。Bob 则拥有了相应的 r_i(选择比特), t_i(某个消息)，如下图所示：

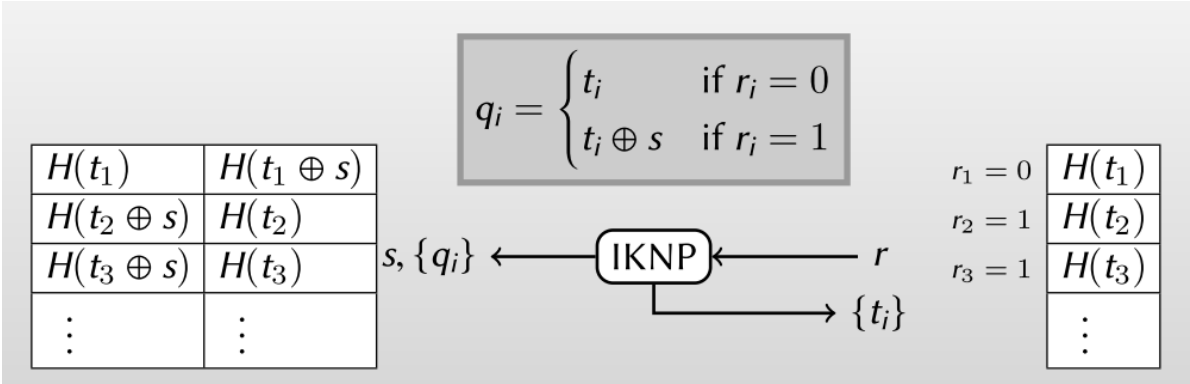


运用上述公式进一步改写为：

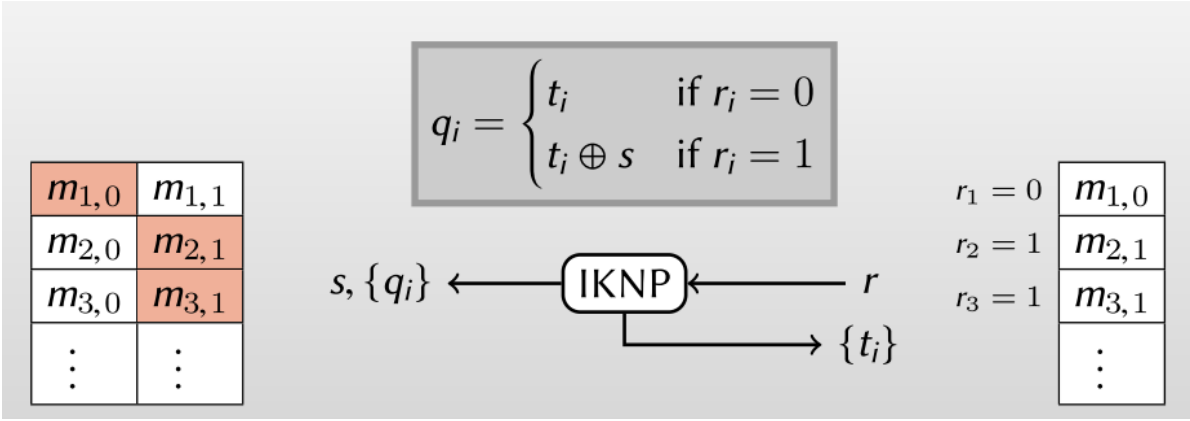


由此就已经可以明显的看出 Alice 拥有了随机的消息对，并且 Bob 拥有选择比特位和 Alice 消息对中的其中一个消息。这已是解决一开始问题（两方获得用于在线阶段的消息队等）。效率方面，扩展矩阵是 $n \times \lambda$ 的矩阵，其中$n \gg \lambda$，而base-OT阶段是按照列来进行，所以会进行 λ 次 base-OT，然而传输了 n-bits 的内容，并且不需要 Yao's GC。

但是仔细观察可以看到，上述的消息对之间重复的使用了同一个比特串 s，这使得生成的消息之间存在相关性，所以必须解决这种下相关性。本协议采用的方法是：使用一个随机语言机来解决相关性。使用的也就是一个随机预言哈希函数 H ，对于消息 $t_1, t_2, \ldots, t_n$ 和比特串 s ， H 使得 $H(t_i, s), H(t_2, s), \ldots, H(t_n, s)$ 是独立伪随机的。从而解决上述问题！整个INKP协议如下图所示：



最后整合为可用的结果为：



此协议是面对半诚实模型的。对于 Bob 恶意的情况下，该模型是危险的！Bob只要再最开始的扩展后的矩阵的一个比特位，就能最后得到 s 的一个比特值，因此只要Bob每一列不同位置的比特值，就能最终恢复出 s 。因此 IKNP 仅仅是半诚实安全。

KK13

在 Kolesnikov，Kumaresan[7]等人 2013 年发表的文章中，对 IKNP03 协议在 GMW 中长度扩展步骤的通信开销远高于核心归约步骤的通信开销问题进行了优化。Kolesnikov等人发现，IKNP03 在base-OT 阶段，对 Bob 的选择比特串 r 进行扩展的时，使用的是重复编码扩展方法，这是最简单的编码方法，其编码效率仅为$\frac{1}{\lambda}$, λ 是扩展后的向量长度。因此Kolesnikov 等人从此着手，使用更加复杂的编码方式在优化 IKNP03 协议。

用 $C(r_i)$ 表示对 r_i 使用某种编码方法 ~~(本文使用的是 Walsh-Hadamard Code)~~，用编码的视角看待 IKNP 协议，首先是 Bob 对选择向量扩展，扩展后的矩阵如下：

r	
1	$\cdots C(1) \cdots$
0	$\cdots C(0) \cdots$
0	$\cdots C(0) \cdots$
0	$\cdots C(0) \cdots$
.	.
.	.

然后再运用秘密分享得到：（其中 t_i 表示 其中一个份额矩阵的行向量）

$$T=\begin{array}{|c|c|}\hline r & \\\hline 1 & \cdots t_1 \cdots \\\hline 0 & \cdots t_1 \cdots \\\hline 0 & \cdots t_1 \cdots \\\hline 0 & \cdots t_1 \cdots \\\hline \vdots & \vdots \\\hline\end{array}\bigoplus\begin{array}{|c|c|}\hline r & \\\hline 1 & \cdots t_1 \cdots \\\hline 0 & \cdots t_1 \cdots \\\hline 0 & \cdots t_1 \cdots \\\hline 0 & \cdots t_1 \cdots \\\hline \vdots & \vdots \\\hline\end{array}$$

通过秘密分享矩阵得到 Alice 的矩阵 Q：

$$Q=\begin{array}{|c|c|}\hline q_1 & q_1 \bigoplus s \\\hline q_2 & q_2 \bigoplus s \\\hline q_3 & q_3 \bigoplus s \\\hline \vdots & \vdots \\\hline\end{array}=\begin{array}{|c|c|}\hline q_1 \bigoplus (C(0)\wedge s) & q_1 \bigoplus (C(1)\wedge s) \\\hline q_2 \bigoplus (C(0)\wedge s) & q_2 \bigoplus (C(1)\wedge s) \\\hline q_2 \bigoplus (C(0)\wedge s) & q_2 \bigoplus (C(1)\wedge s) \\\hline \vdots & \vdots \\\hline\end{array}$$

使用 $q_i = t_i \bigoplus (C(r_i)\wedge s_i)$,改写 Q 矩阵为：

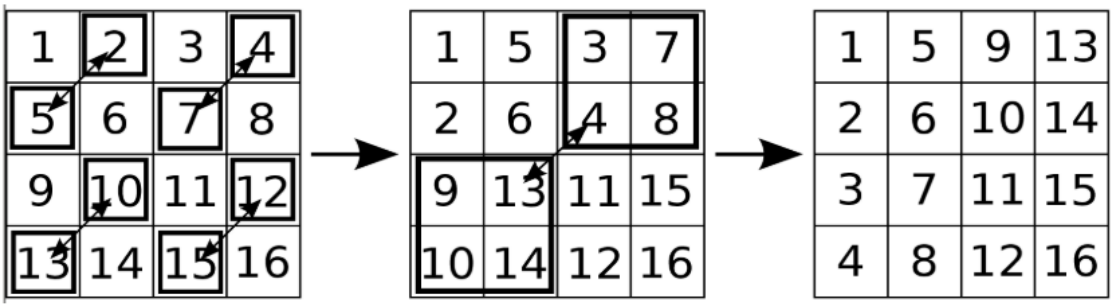
$$\begin{array}{|c|c|}\hline t_1 \bigoplus (C(1)\wedge s)\bigoplus (C(0)\wedge s) & t_1 \bigoplus (C(1)\wedge s)\bigoplus (C(1)\wedge s) \\\hline t_2 \bigoplus (C(0)\wedge s)\bigoplus (C(0)\wedge s) & t_2 \bigoplus (C(0)\wedge s)\bigoplus (C(1)\wedge s) \\\hline t_3 \bigoplus (C(0)\wedge s)\bigoplus (C(0)\wedge s) & t_3 \bigoplus (C(0)\wedge s)\bigoplus (C(1)\wedge s) \\\hline \vdots & \vdots \\\hline\end{array}=\begin{array}{|c|c|}\hline t_1 \bigoplus (C(1)\wedge s) & t_1 \\\hline t_2 & t_2 \bigoplus (C(1)\wedge s) \\\hline t_3 & t_3 \bigoplus (C(1)\wedge s) \\\hline \vdots & \vdots \\\hline\end{array}$$

可以看到最后 Alice 获得结果进一步变为：$t_i, t_i \bigoplus (C(1)\wedge s)$，然后再用随机预言机来破坏其关联性：$H(t_i), H(t_i \bigoplus (C(1)\wedge s))$ Bob的值也就变为 $H(t_1), H(t_2), \ldots, H(t_n)$。

从编码的方式了解了 IKNP 之后就可以使用其他的编码方式对其推广了，相比于 IKNP 也就是 Alice 获得的两个消息结果中的编码部分改变而已，例如用 ε_i 表示一个编码结果，则 Alice 获得的两个消息变为 $H(t_i), H(t_i \bigoplus (C(r_i \bigoplus \epsilon)\wedge s))$。而 Bob 的结果并不变。所以改进后协议的提升也都来自编码效率的改变。

ALSZ13

ALSZ13[8] 在 IKNP 基础上多通信复杂度和计算复杂度都进行了优化，首先是算法方面的计算复杂度，Asharov 经过实验发现 IKNP 协议中大约 42% 的计算耗费在矩阵转置上，于是对于矩阵给的转置进行优化，如下图所示：



先对最小的 2×2 子矩阵进行转置，只需要消耗很少的时间，进而再将该 2×2 矩阵看为整体再寻找下一个“2×2” 子矩阵。并且因为转置之间不冲突，再同一个层级上可以并行。如此一来将 m×n 的矩阵转置的计算消耗从 $O(mk)$ 降低为 $O(m/rlog_2 k)$。其中 r 是CPU 寄存器大小。此外因为 Bob 拥有的 t_i 组成的矩阵列之间没有相关性，所以可以在 OT 中使用并行算法加速。在通信复杂度方面，Asharov 等人使

用盲化因子 $s_{j,0}$ 来初始化 Bob 持有的矩阵而不是 随机生成，这样每次只需要向 Alice 传输一个消息就可，具体如下。原本的协议是：(其中PRG()是伪随机序列生成器)

$$
\begin{array}{ll}
 & \mathbf{T} \in_R \{0,1\}^{m \times k} \\
 & \text{for } 1 \le j \le k: \\
 & \quad u_{j,0} = PRG(s_{j,0}) \oplus \mathbf{T}[\mathrm{j}] \\
\text{for } 1 \le j \le k: \quad (u_{j,0}, u_{j,1}),\ 1 \le i \le k & \quad u_{j,1} = PRG(s_{j,1}) \oplus \mathbf{T}[\mathrm{j}] \oplus \mathbf{r} \\
\quad \mathbf{V}[\mathrm{j}] = u_{j,c_j} \oplus PRG(s_{j,c_j}) &
\end{array}
$$

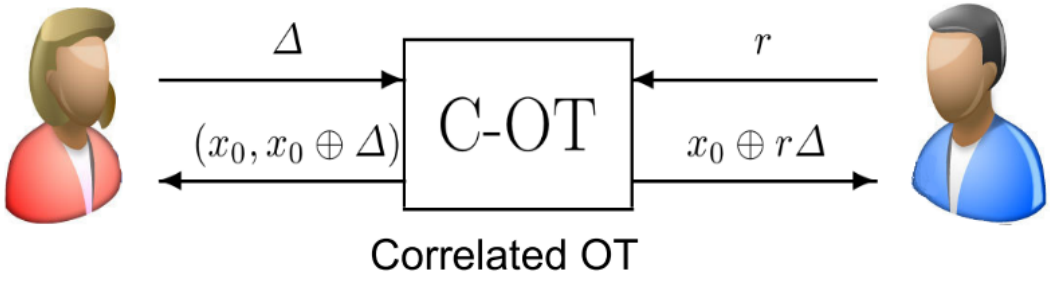
优化后：

$$
\begin{array}{ll}
 & \text{for } 1 \le j \le k: \\
 & \quad \mathbf{T}[\mathrm{j}] = PRG(s_{j,0}) \\
 & \quad u_j \quad = PRG(s_{j,1}) \oplus \mathbf{T}[\mathrm{j}] \oplus \mathbf{r} \\
\text{for } 1 \le j \le k: \quad u_j,\ 1 \le i \le k & \\
\quad \mathbf{V}[\mathrm{j}] = c_j u_j \oplus PRG(s_{j,c_j}) &
\end{array}
$$

在 IKNP 中有： $q_i = t_i \bigoplus (r_i \wedge c_i)$ ，以此来验证一下上述优化的正确性，当 $r_i \wedge c_i = 0$ 时，易得 $V[j] = T[j]$；当$r_i \wedge c_i = 1$ 时，易得$V[j] = T[j] \bigoplus r$ ，这符合 IKNP 中的结论。并且可以看出，该优化方法理论上可以降低此过程一半通信消耗。

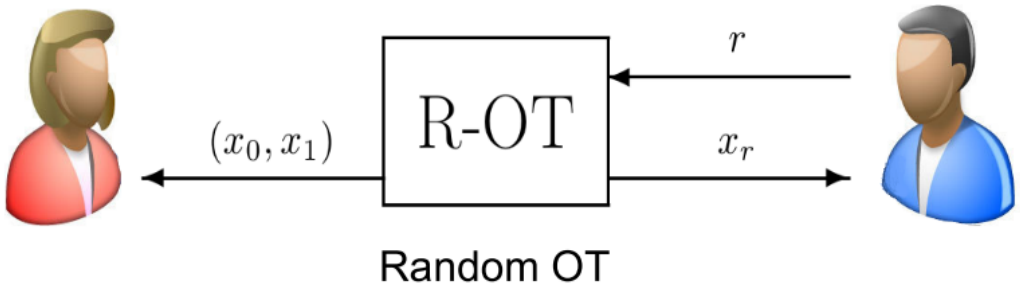
Asharov 等人还对应用在 Yao'S GC 和 GMW 上的 OT-Extension 做出了相应的优化。

在 Yao'S GC 上，对于最后一部分的 OT ，使用了Correlated-OT 来降低带宽，C-OT如图：



x_0 是一个随机比特，$x_1 = x_0 \bigoplus \Delta$,其中 Δ 是一个全局随机量，r 是Bob 的选择比特，在之前叙述的通用 OT-Extension 中，Alice 接下来需要计算并发送两个消息 $y_i^0 = x_i^0 \bigoplus H(q_j), y_i^1 = x_i^1 \bigoplus H(q_i \bigoplus s)$ 。但是在应用了上述C-OT 的 Yao'S GC 上只需要Alice 设定 $x_i^0 = H(q_i)$ ，计算并发送一个消息 $y_i = \Delta \bigoplus H(q_i) \bigoplus H(q_i \bigoplus s)$ 即可。而 Bob 根据自己的 r 值来得到自己的输出，当$r_i = 0$,得到 $H(t_i)$ ； 当$r_i = 1$，得到 $H(t_i) \bigoplus y_i$ 。很容易证明，不同的 r 值会让 Bob 得到 x_0 *or* x_1 ，所以此方法可行，并且因为只需要传输一个消息，则降低了一半的带宽需求。

在 GMW 上则使用了Random-OT 来优化，如图：



其中 Alice 获得的 (x_0, x_1) 都是随机的，之后 Alice 设定 $x_0 = H(q_i), x_1 = H(q_i \bigoplus s)$,而后 Bob 只需要计算 $b^{r_i} = H(t_i)$ ， 因为$q_i = t_i \bigoplus (r_i \wedge c_i)$ ， xianran 当$r_i = 0$,得到 $H(t_i) = H(q_i)$ ； 当$r_i = 1$，得到 $H(t_i) = H(g_i) \bigoplus y_i$ 。如此一来不需要进行最后一步的通信就可以得到消息。

参考文献

[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] Impagliazzo R ， Rudich S . Limits on the provable consequences of one-way permutations (invited talk). Springer New York, 1990.

[5] Beaver D . Correlated Pseudorandomness and the Complexity of Private Computations[C]// Twenty-eighth Acm Symposium on the Theory of Computing. ACM, 1996.

[6] Ishai Y ， Kilian J ， Nissim K , et al. Extending Oblivious Transfers Efficiently[C]// 23rd Annual International Cryptology Conference. CiteSeer, 2003.

[7] Kolesnikov V ， Kumaresan R . Improved OT Extension for Transferring Short Secrets[M]. Springer Berlin Heidelberg, 2013.

[8] Asharov G ， Lindell Y ， Schneider T , et al. More efficient oblivious transfer and extensions for faster secure computation[C]// Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security. ACM, 2013.