MPC 理解

张聪

2021 年 11 月 5 日

1 MPC 的主要方法

目前 MPC 的主要方法有：Garbled Circuit [Yao86]，GMW [GMW87]，BGW [BGW88]，BMR [BMR90]。下面分别介绍。

- GC: 最早由 [Yao86] 提出，其特点是轮数为常数，通信量大，且只能用于两方计算。推荐阅读 [LP04] 作为 GC 协议的入门。由于 GC 最初是作为一个协议提出的，[BHR12] 将 GC 作为一个密码学组件进行了抽象，将 GC 方案分为 (Garble, Encode, Eval, Decode) 几个算法，并定义了 Privacy，Obliviousness，Authenticity。

  值得注意的是，GC 天然地只适用于布尔电路，即每条线只有 0/1 两个值，如果平凡地推广到算数电路 $\mathbb{F}_p$ 上，那么 garbled table 则有 p^2 行，当 p 很大时，这是不可接受的。[AIK11] 最早考虑了 garble 算数电路的情况，其思路是对输入进行编码，本质上还是布尔电路，后来 [BMR16] 则第一次考虑了推广 GC 到算数电路的情况，每条线有 p 个 label，借鉴半门的思想，每个乘法门可以降低到 $2p$ 条密文。后续发展有 [Ben18, BCM+19, MW19]。

  - 半诚实情况：作为 GC 后续发展，相继提出了点置换技术 (point-and-permute) [BMR90]，Free-XOR 技术 [KS08]，行约减技术 (grable row reduction,GRR)[NPS99, PSSW09]，半门技术 (half-gate) [ZRE15]，分片切割技术 (slicing-and-dicing) [RR21]. 其中 state-of-the-art 是 [RR21]，可以做到每个 AND 门通信量 $1.5\kappa + 5$ 比特，同时与 FreeXOR 兼容。
  - 恶意情况：值得注意的是，最初的 GC 协议其实对 malicious 的 evaluator 也是安全的。我理解主要是因为 GC 协议给 garbler 的“权利”太大，garbler 可以完全主导 GC 的生成，因此即使他想生成一个错误的电路，evaluator 也没办法察觉。而 evaluator 能做的仅仅是拿到 label 之后本地求值，无法搞什么恶意破坏。因此恶意 GC 主要就是要防止恶意的 garbler。

    我所接触的最早文章是 Lindell 和 Pinkas 的 [LP07]，其主要思想是使用 cut-and-choose 方法，即让 garbler 一次性生成多个 GC，然后让 evaluator 随机选择一些打开验证是否是正确的电路。其后续发展有 [LP11, Lin13, LR14, LR15, RR16, WMK17],

其主要思路都是类似的，区别在于具体如何提高效率，减少多余电路的传输，减少公钥操作的使用，batch 和 online/offline 的优化等。可以参考我之前的 MPC 总结。另外一种思路是由 Xiao Wang 等人 [WRK17a] 提出的认证 garble 思路。该协议的主要思路是，其实我们只需要保证 garbler 生成 GC 是诚实生成的就好了，进一步来说，由于 FreeXOR，因此只需要保证 AND 门的 garbled table 是诚实生成的就行了，也就是 AND 门的真值表要是真实的。又由于 garbled table 的排列和置换比特相关，要想不让 garbler 捣乱，需要对这个置换比特做认证，即做一次信息论 MAC。使得 AND 门的置换比特是两方的分享，而不是完全由 garbler 控制，且 evaluator 有 MAC 密钥可以验证正确性。此外这里面还用到一个思想是信息论 MAC 里有一个全局密钥 Δ，而这个 Δ 恰好可以用作 FreeXOR 里的全局差分，这样生成的 GC 其实也是一个两方分享的形式，garbler 失去了对 GC 完全的控制权。后续的发展有 [KRRW18]，主要把 half-gate 的思路用到了认证 GC 上，降低通信。

- GMW：最早由 [GMW87] 提出，其特点是通信轮数与电路深度成正比，但通信量低。如果 GC 是最早的两方解决方案，那么 GMW 可以看成是最早的多方解决方案。他们的思路完全不同。GMW 的主要思路是各方将自己的输入进行加法秘密分享，即 $x = x_1 + \cdots + x_n$（这里可以是布尔值也可以是算数值），然后对于加法门，各方直接本地把分享加起来即可，对于乘法门，则主要有两种方法进行计算：OT/OLE，HE。如果考虑 offline 情况，可以使用 Beaver triple。最早的 GMW 协议考虑的是布尔电路，他们用的 4 选 1OT 来做乘法，具体可参考 Goldreich 的书 [Gol04]。
  - 半诚实情况：据我所知似乎没有改进半诚实协议的文章，现在最快的半诚实 GMW 协议依然是最初的版本 [GMW87]。
  - 恶意情况：恶意版本的 GMW 在最初的文章 [GMW87] 就已经提出了，即 GMW 通用编译器，其主要思路是对每一条发出的消息都进行一次零知识证明，证明这条消息是按照协议规范计算的。其优点是对任何协议都通用，缺点则是效率非常低。后面的主要发展主要有 [BDOZ11, DPSZ12]，后者就是著名的 SPDZ 协议。其主要思想是各方使用信息论 MAC 对每条线上的秘密分享做认证，在输出打开之前先验证 MAC，验证通过了再输出。区别在于 BDOZ 式认证中，对每条线上的真值的每一对分享进行两两认证（即每对 P_i 和 P_j 的秘密均进行认证），而 SPDZ 式认证是每方有一个全局密钥的分享和每一条线上真值的 MAC 的分享。这两篇文章生成认证 MAC 的方式都是使用半同态加密，而 [NNOB12] 和 [KOS16] 使用 OT 代替半同态加密构造认证 MAC，分别拓展了 BDOZ 和 SPDZ 式认证。其中使用 OT 构造 BDOZ 式认证的方法也称为 TinyOT 协议，关于 TinyOT 的详细脉络可以参考 [BLN+15](TinyOT 除了在这里有用之外，在 Xiao Wang 的一系列多方恶意 GC 协议的构造中也大有用途)。而 SPDZ 后面也有很多发展，有 [DKL+13, KOS16, CDE+18, KPR18, KOS16, Kel20]，这方面的文章我看的不多，

就不做过多描述了。

- BGW：最早由 [BGW88] 提出，其特点是通信轮数与电路深度成正比，通信量低。这里 BGW 和 GMW 最大的区别是，GMW 使用加法分享，而 BGW 使用 Shamir 分享。在我看来，加法分享可以看成特殊的门限分享，即 $(n-1,n)$ 分享，因此 GMW 可以允许 dishonest majority，即使敌手 corrupt 了 $n-1$ 个人，依然无法得到诚实方的输入。而 BGW 使用的 Shamir 秘密分享的 corrupt 门限为半诚实 $t<\frac{n}{2}$，恶意 $t<\frac{n}{3}$，即 honest majority。虽然 BGW 可抵抗的敌手人数降低了，但 BGW 可以实现信息论安全，而 GMW 只能实现计算安全 (这里的理解是由于 $t<\frac{n}{2}$，各方要做乘法可以直接将秘密分享本地做乘法，得到 $2t$ 的秘密分享，接下来再经过一个次数约减步骤，将这个 $2t$ 次分享降为 t 次分享，这个步骤是可以直接再使用秘密分享来做，不需要任何假设。而 GMW 无法这么做，必须使用如 OT，HE 等计算组件来完成，因此是计算安全的)，且 BGW 协议可以保证输出呈递性 (Guaranteed Output Delivery,GOD) 或公平性 (fairness)，而 GMW 只能保证可终止安全性 (security with abort)。这里可以理解为安全性的一个 trade-off，能抵抗的敌手能力越强，则能抵抗的敌手数量越少。关于 BGW 协议的一个全面描述与安全证明，推荐阅读 [AL17]。
  值得一提的是 BGW 协议可以进一步推广到一般的线性秘密分享方案 (Linear Secret Sharing Scheme,LSSS)[CDM00]，该文章定义了可乘性 (multiplicative) 与一般的敌手结构 (adversary structure)，考虑在更一般的情况下如何用一个抽象的秘密分享方案构造 MPC。这篇文章比较抽象，我了解的不多。但是这里的抽象给出了使用别的秘密分享方案的可能，其中一个比较著名的秘密分享就是重复秘密分享 (Replicated Secret Sharing,RSS)[CDI05]，这篇文章考虑了用 RSS 构造 MPC，并且给出了用 RSS 转化成 Shamir 秘密分享的方法，即伪随机秘密分享 (PRSS)。后面我会描述。一般提到 BGW 协议还是默认使用 Shamir 秘密分享。

  - 半诚实情况：在最初的 BGW 协议中，乘法协议是让各方将分享本地相乘之后执行一个次数约减子协议，这个次数约减子协议需要各方将自己的秘密分享再做一次秘密分享，因此通信复杂度为 $O(n^2)$。随后 [DN07] 提出了新的乘法协议，其思路是选出一个代表方 P_{king} 收集各方随机化之后的 $2t$ 分享，重构之后在用 t 次分享回去，这样通信复杂度就是 $O(n)$，具体来说，每个乘法门需要通信 6 个有限域元素。目前最新的 [GLO+21] 通过降低随机对的生成成本，将每个乘法门通信降低到了 4 个域元素。
  - 恶意情况：最早的 BGW 就提出了恶意情况的协议，主要思路是使用可验证秘密分享 (Verifiable Secret Sharing,VSS)，保证敌手无法分享错误的秘密，且将敌手门限降低到了 $\frac{n}{3}$，由于 Shamir 分享可以看成 RS 编码，利用纠错码的性质，即使敌手搞破坏，诚实方也可以恢复出正确的输出。[GIP+14] 发现很多半诚实乘法协议 (如 DN 协议) 其实在恶意敌手情况下，敌手除了可以做一种加法攻击外，也是安

全的 (secure up to an additive attack)，即敌手最多只能在乘法结果上多加上某个数，因此恶意安全的协议只需要在半诚实协议基础上，在最后结果被打开之前增加一步验证步骤即可。验证协议目前的发展主要有 [BFO12, LN17, FLNW17, NV18, CGH+18, FL19, GS20, BGIN19, BGIN20]

- BMR：最早由 [BMR90] 提出，其实就是 GC 的多方版本，因此其特点和 GC 是一样的，即常数轮，但通信量大。由于在两方 GC 协议中，garbler 的权利是非常大的，因此想要推广到多方，谁来做 garbler 就成了一个问题。BMR 的解决方法是，让所有参与方共同使用一个 MPC 协议来生成 GC(分布式生成 GC)，再指定一个求值者求值即可。BMR 框架比较关键的就是如何设计分布式生成 GC 的子协议，可以参考 [EKR18] 这本书的 3.5 节了解。
  - 半诚实情况：半诚实情况和 GC 的发展比较少，[BLO16, BLO17] 将 FreeXOR 技术推广到多方 GC 中，且使用密钥同态 PRF 降低了渐进通信复杂度。
  - 恶意情况：恶意情况的构造比较多，[LPSY15, LSS16] 分别使用 SPDZ 和 SHE 对每个 garbled table 做认证，[HSS17, WRK17b, YWZ20] 则是使用 TinyOT 协议做认证，目前 state-of-the-art 是 [YWZ20] 的结果，将半门的思想用在了多方 GC 中，且改进了 TinyOT 的生成效率。

1.1 安全定义

目前 MPC 的安全性根据不同的场景有很多种分类方式，大致有下面几种：

- 按照敌手行为划分：
  - 半诚实/被动 (semi-honest/passive)，即敌手按照协议规定执行，想从协议的执行副本中获得额外的信息
  - 恶意/主动 (malicious/active)，即敌手可以任意偏离协议规范执行。
  - 隐蔽 (covert)：由 [AL07] 最早提出，即敌手表现为恶意时有一定的概率 ϵ 被诚实方发现，这里 ϵ 被称为威慑因子 (deterrence factor)。当威慑因子为 overwhelming 时，covert 和 malicious 就是等价的。随后 [AO12] 在此基础上提出了更高的安全性，称为公开可验证的隐蔽安全性 (publicly verifiable covert, PVC)，表示敌手一旦被抓到作弊，诚实方可以生成一个证明 π，可以证明敌手作弊，且可以由任何人公开验证。目前 PVC 的后续工作比较多，有 [KM15, HKK+19, DOS20, FHKS21, SSS21] 前两篇是提升效率，后几篇是通用编译器。
- 按敌手计算能力划分：
  - 信息论/完美/无条件安全 (information theory/perfect/unconditionally security)，即模拟器的输出和真实的 view 同分布。

 - 统计安全 (statistical security)，即模拟器的输出和真实的 view 统计不可区分。
 - 计算安全 (computational security)，即模拟器的输出和真实的 view 计算不可区分。

- 按敌手数量划分：
 - 不诚实大多数（dishonest majority）：即敌手最多可以有 $n-1$ 个。用 t 表示敌手数量，则 $t<n$。目前基于 GC，GMW，BMR 框架的协议均满足这一点。两方协议一定是 dishonest majority。
 - 诚实大多数（honest majority），即敌手的数量不超过参与方总数的一半，$t<\frac{n}{2}$。主要在 BGW 框架下。有时为了达到更高的安全性或者效率会有 $t<\frac{n}{3}, t<(\frac{1}{2}-\epsilon)n$ 等情况。
- 按敌手对输出的控制程度划分：
 - 保证输出呈递性 (guaranteed output delivery,GOD)，即敌手无法干扰诚实方得到正确的输出。这个性质是最强的，但并不总是能够保证，只有 honest majority 可以保证 GOD [GSZ20]。
 - 公平性 (fairness)，即敌手要么和诚实方同时得到输出，要么同时终止。GOD 可以推出 fairness，但反之不成立。同样地，fairness 也并不总是能够保证，例如永远无法实现公平的投币协议 [Cle86]。
 - 可终止安全性 (security with abort)，即敌手可以先得到输出，再由敌手决定是否让诚实方得到输出。这也是目前实际 MPC 构造中最常见的类别。可终止安全性又可以分为下面几类：
 * Security with identifiable abort，即诚实方可以知道敌手身份。
 * Security with unanimous abort，即诚实方可以检测到敌手但不知身份，敌手可以选择让所有诚实方同时终止或输出。
 * Security with selective (non-unanimous) abort，即诚实方要么检测到作弊行为而终止要么输出，敌手可以选择让哪些诚实方输出，哪些诚实方终止。一般只说 security with abort 时默认指的就是这种情况。

 [CGZ20] 研究了广播轮数和这些可终止安全性之间的关系。
- 按敌手确定时间划分：
 - 静态 (static)：即敌手在协议开始前就确定 corrupt 哪些 party。目前绝大部分实用的 MPC 协议都是考虑静态敌手。
 - 动态 (adaptive)：即敌手可以在协议开始后再确定 corrupt 哪些 party。这里 corrupt party 一旦被 corrupt 后就一直是 corrupt party。根据敌手是否能读取 corrupt 方过去的 view，又可以分为两类：

    * No erasures model，即敌手在 corrupt 之后可以得到 corrupt party 的整个 view，包括他的输入，随机带，收到的消息。[CFGN96, Can00, CLOS02]
    * Erasures model，即允许 corrupt party 在被敌手 corrupt 时擦除一些数据。[BH92, Lin09]
  - 移动 (proactive/mobile)：即敌手可以只 corrupt 某个 party 一段时间，也就是说，honest party 可能变成 corrupt party，而 corrupt party 也可能变成 honest party。[OY91, CH94]

2 半诚实乘法协议总结

由上个部分可以看到，在 GC 框架下的协议有 FreeXOR，因此布尔域上的加法是不需要通信的，唯一需要通信的就是 AND 门，即乘法。而 GMW 和 BGW 均使用秘密分享，对加法门来说也只需要各方自己本地计算输入分享的加法即可，需要通信的也只有乘法门。因此，MPC 发展的本质就是如何更好地做乘法。下面总结一下目前我所知道的做乘法的方法，这里只考虑半诚实协议，不考虑恶意，因为半诚实情况下的思路更直接，就是纯粹只做乘法，而恶意协议一般是考虑怎么加入一些验证步骤防止敌手作弊。这里不考虑验证。后面我会再考虑总结一下目前已有的乘法验证协议。

2.1 重复秘密分享

在介绍乘法协议之前，先介绍一下重复秘密分享 (Replicated Secret Sharing,RSS)，这种分享的特点是每个人拿到的秘密分片是指数大小 (C_{n-1}^t 个)，但好处是可以不需要交互地转化成 Shamir 秘密分享，且用来做乘法只需要通信一个元素。

RSS 方案也是一个门限方案，设 (n,t)-RSS 方案表示 n 方分享的 $t+1$ 门限方案，即 t 个人无法恢复秘密，$t+1$ 个人才可以恢复秘密。RSS 的思路是把秘密写成加法分享，分成 C_n^t 项，每一项的下标就是 C_n^t 中的一种组合，然后每个人 P_i 拿下标为自己不属于那个组合的分片，即每人拿 C_{n-1}^t 个项。此时任意 t 个人无法恢复秘密，因为他们拿到的项一定会差一个下标为这 t 个人的项。

这里举一个例子 $(n=5,t=2)$：要分享秘密 x，先写成加法形式，

$$x = x_{12} + x_{13} + x_{14} + x_{15} + x_{23} + x_{24} + x_{25} + x_{34} + x_{35} + x_{45}$$

秘密分享如下:

$$P_1 : x_{23}, x_{24}, x_{25}, x_{34}, x_{35}, x_{45}$$
$$P_2 : x_{13}, x_{14}, x_{15}, x_{34}, x_{35}, x_{45}$$
$$P_3 : x_{12}, x_{14}, x_{15}, x_{24}, x_{25}, x_{45}$$
$$P_4 : x_{12}, x_{13}, x_{15}, x_{23}, x_{25}, x_{35}$$
$$P_5 : x_{12}, x_{13}, x_{14}, x_{23}, x_{24}, x_{34}$$

可以看到，让每个 P_i 拿到的是下标里不含 i 的那些项，此时如果想要恢复秘密，需要至少 3 个人合作，因为任意两个人 P_i, P_j 拥有的项一定不包括 x_{ij}，所以无法恢复秘密。

注：有的 RSS 方案会描述为秘密的下标是上述的补集，即 C_n^{n-t} 中的组合，此时 P_i 拿到的项就是下标包含 i 的项。任意 t 方依然无法恢复秘密，这是因为他们一定缺少一项下标为他们之外 $n-t$ 个人的那一项。由 $C_n^t = C_n^{n-t}$，这两种方案是一样的，只是看待的角度不同。

2.1.1 分享转换，伪随机秘密分享

当各方拥有 RSS 时，可以通过本地直接转换成 Shamir 秘密分享。设 (n, t)-RSS 方案为

$$x = \sum_{A \subset [n] : |A| = t} x_A$$

想要将 x 转换为 $(n, t)-$Shamir 分享，考虑对每个 $A \subset [n], |A| = t$ 构造 t 次多项式 f_A，满足

1. $f_A(0) = 1$,
2. $f_A(i) = 0$ for $i \in A$

此时每一方 P_j 计算自己的分享为

$$x_j = \sum_{A \subset [n] : |A| = t, j \notin A} x_A \cdot f_A(j)$$

这些 $\{x_j\}_{j \in [n]}$ 构成了 $(n, t)-$Shamir 分享。这是因为，考虑 t 次多项式

$$f = \sum_{A \subset [n] : |A| = t} x_A \cdot f_A$$

则 $f(j) = x_j$ 且 $f(0) = \sum_A x_A = x$

上述方法只能将一个 RSS 转换成一个 Shamir 秘密分享。一个观察是，当秘密 x 是随机的时候，所有的 RSS 分片 x_A 是独立随机的。此时可以将 x_A 作为伪随机函数 $\psi(\cdot)$ 的密钥，只要各方对一个常数 a 达成共识，则可以直接用 $\psi_{x_A}(a)$ 代替 x_A 构造 Shamir 分享，即:

$$x_j = \sum_{A \subset [n] : |A| = t, j \notin A} \psi_{x_A}(a) \cdot f_A(j)$$

这样，只要各方有了一个 RSS，就可以本地计算无穷个随机 Shamir 秘密分享。需要注意的是这里的秘密分享其实和真正的 Shamir 分享不太一样，真正的 Shamir 分享的秘密分片是真随机的，这里的秘密分片是伪随机的 (即，伪随机秘密分享，PRSS)，这会导致用这种方法得到的 MPC 不再是信息论安全，而是计算安全的了。

2.2 OT 与 OLE

2.2.1 OT 与 OLE 的关系

在讲乘法之前，再讲一下我对 OT 与 OLE 的关系的理解。OLE 其实就是 OT 在算数域上的推广。在 OT 中，sender 的输入是 (m_0, m_1)，receiver 输入 $x \in \{0,1\}$，输出 m_x。在 OLE 中，sender 的输入是 (a,b)，receiver 输入 $x \in \mathbb{F}$，输出 $ax+b$，这里 $\mathbb{F}$ 是一个有限域。从另一个角度看 OT 的输出，$m_x = m_0 \oplus x \cdot (m_0 \oplus m_1)$，如果我们转换一下 sender 的输入，令 $a := m_0 \oplus m_1, b := m_0$，此时 OT 的输出就是 $ax \oplus b$，和 OLE 形式就一样了。

2.2.2 乘法分享协议

MPC 里面我们最关注的就是如何把乘法转化成加法分享，即：sender 输入一个 $x \in \mathbb{F}$，receiver 输入 $y \in \mathbb{F}$，sender 输出 $m \in \mathbb{F}$，receiver 输出 $n \in \mathbb{F}$，满足 $m+n = xy$。称这个协议为乘法分享协议 (multiplication sharing, MS)。

OT/OLE 其实和乘法分享协议等价，即可以互相构造。这里以更一般的 OLE 为例：

OLE $\rightarrow$ MS：

1. receiver 随机选取 m，令 $a := y, b := -m$。
2. 双方调用 OLE，sender 输入 x，receiver 输入 a, b。sender 得到 $n = ax+b = yx-m$，满足 $m+n = xy$。

MS $\rightarrow$ OLE：

1. sender 和 receiver 对 a, x 调用 MS 得到 ax 的分享 m, n，满足 $ax = m+n$。
2. sender 将 $c = b+m$ 发送给 receiver。
3. receiver 计算 $n+c = n+b+m = ax-m+b+m = ax+b$。

2.2.3 用同态加密做乘法分享

如果我们有同态加密，则可以非常简单地做乘法分享，协议如下：

1. sender 生成 HE 的公私钥对 (pk, sk)，计算 $c = Enc_{pk}(x)$ 发给 receiver。
2. receiver 随机选一个 r，利用同态性质，计算 $y \cdot c + r = Enc_{pk}(xy+r)$ 发回 sender。
3. sender 解密得到 $xy+r$ 作为自己的分享，receiver 令 $-r$ 作为自己的分享。

2.3 GMW 乘法协议

GMW 中各方有加法分享。设参与方有 n 个，分别为 $P_1, \ldots, P_n$。电路中每条线上的真值为 x，则每方有一个加法分享，即 P_i 有 x_i，满足 $x = x_1 + \cdots + x_n$。一般用 $[x], [y], [z]$ 表示加法分享，即 $[x] = (x_1, \ldots, x_n)$ 满足 $x_1 + \cdots + x_n = x$。乘法协议是指各方持有乘法门的输入分享，想求乘法门的输出分享，即各方持有 $[x], [y]$，想求 $[z] = [xy]$。

2.3.1 两方情况

这里先介绍两方布尔情况，即 P_1 有 x_1, y_1，P_2 有 x_2, y_2，双方想求 z_1, z_2 满足 $z_1 \oplus z_2 = (x_1 \oplus x_2) \cdot (y_1 \oplus y_2)$。一个思想是使用上一节的 MS 协议。等式右边展开有 $x_1y_1 \oplus x_1y_2 \oplus x_2y_1 \oplus x_2y_2$，这里 x_1y_1, x_2y_2 可以分别由 P_1 和 P_2 本地计算，而交叉项 x_1y_2 和 x_2y_1 恰好可以用上一节的 MS 协议来得到加法分享。协议如下：

1. P_1 用 x_1, y_1 和 P_2 用 y_2, y_1 调用两次 MS 协议。P_1 得 m_1, m_2，P_2 得 n_1, n_2，满足 $m_1 \oplus n_1 = x_1y_2$，$m_2 \oplus n_2 = x_2y_1$
2. P_1 令 $z_1 := x_1y_1 \oplus m_1 \oplus m_2$，$P_2$ 令 $z_2 := x_2y_2 \oplus n_1 \oplus n_2$。则 $z_1 \oplus z_2 = x_1y_1 \oplus m_1 \oplus m_2 \oplus x_2y_2 \oplus n_1 \oplus n_2 = x_1y_1 \oplus x_1y_2 \oplus x_2y_1 \oplus x_2y_2 = (x_1 \oplus x_2) \cdot (y_1 \oplus y_2)$。

另外一种思路是不展开，既然是在布尔域上，虽然 P_1 不知道 P_2 的输入，但是可以直接遍历，总共就 4 种情况，因此可以直接调用一次四选一 OT 来实现。协议如下：

1. P_1 选一个随机比特 $r \leftarrow \{0, 1\}$。
2. P_1 和 P_2 调用一次四选一 OT，P_1 作为 sender，输入是 $(x_1y_1 \oplus r, x_1 \cdot (y_1 \oplus 1) \oplus r, (x_1 \oplus 1) \cdot y_1 \oplus r, (x_1 \oplus 1) \cdot (y_1 \oplus 1) \oplus r)$，$P_2$ 作为 receiver，输入 $2x_2 + y_2 + 1$(即 x_2y_2 对应的二进制数字)。P_2 得到输出 $(x_1 \oplus x_2) \cdot (y_1 \oplus y_2) \oplus r$。
3. P_1 令 $z_1 := r$，P_2 令 $z_2 := (x_1 \oplus x_2) \cdot (y_1 \oplus y_2) \oplus r$。显然 $z_1 \oplus z_2 = (x_1 \oplus x_2) \cdot (y_1 \oplus y_2)$。

上面协议中选 r 的作用是生成秘密分享，如果不加上 r，会让 P_2 直接拿到乘法结果，而不是分享。

上述两种协议如果推广到有限域 $\mathbb{F}_p$ 上，第一个协议只需要把 OT 换成 OLE 即可，第二个协议则只需把四选一 OT 换成 $2p$ 选一 OT 即可。

2.3.2 多方情况

多方情况的 GMW 可以转化为两方情况。在多方情况下，P_i 有 x_i, y_i，满足 $\sum_{i\in[n]} x_i = x, \sum_{i\in[n]} y_i = y$，$P_i$ 想要得 z_i 满足 $\sum_{i\in[n]} z_i = (\sum_{i\in[n]} x_i)(\sum_{i\in[n]} y_i)$。有如下观察：

$$
\begin{aligned}
(\sum_{i\in[n]} x_i)(\sum_{i\in[n]} y_i) &= \sum_{i\in[n]} x_i y_i + \sum_{1\le i<j\le n} (x_i y_j + x_j y_i) \\
&= (1-(n-1))\cdot \sum_{i\in[n]} x_i y_i + \sum_{1\le i<j\le n} (x_i + x_j)\cdot(y_i + y_j) \\
&= (2-n)\cdot \sum_{i\in[n]} x_i y_i + \sum_{1\le i<j\le n} (x_i + x_j)\cdot(y_i + y_j)
\end{aligned}
$$

可以看到，$(2-n)x_i y_i$ 可以由 P_i 本地计算，后面的项恰好对应于 P_i, P_j 的两方乘法协议。因此调用 C_n^2 次两方乘法协议即可。

2.4 BGW 乘法协议

BGW 协议和 GMW 不一样，使用的是 Shamir 分享，因此上面的方法不太适用。实际上 BGW 协议是无条件安全的，不需要 OT 或 HE 来计算，各方仅需通信即可完成乘法。

设参与方有 n 个，分别为 $P_1, \ldots, P_n$。电路中每条线上的真值为 x，则每方有一个 Shamir 分享，即 P_i 有 x_i，满足存在一个 t 次多项式 f，使得 $f(i) = x_i$ 且 $f(0) = x$。这里 $t < \frac{n}{2}$ 表示敌手数量。一般用 $\langle x\rangle_t, \langle y\rangle_t, \langle z\rangle_t$ 表示 t 次 Shamir 分享 (门限为 $t+1$)。乘法协议是指各方持有乘法门的输入分享，想求乘法门的输出分享，即各方持有 $\langle x\rangle_t, \langle y\rangle_t$，想求 $\langle z\rangle_t = \langle xy\rangle_t$。

设分享 x 的多项式是 f，分享 y 的多项式是 g，即 $x_i = f(i), x = f(0), y_i = g(i), y = g(0)$。Shamir 分享的一个好处是，各方可以本地直接把自己的分享相乘，即 P_i 令 $z_i := x_i y_i$。可以发现 $z_i = f(i)g(i)$，不妨设 $h = fg$ 是 $2t$ 次多项式，则显然 $h(0) = f(0)g(0) = xy$。因此这里 z_i 是 xy 的 $2t$ 次分享。这里有两个问题，首先，z_i 其实不是标准的 Shamir 分享，因为标准的 Shamir 分享要求这个多项式是随机选取的，但是这里 h 是由两个 t 次多项式相乘得到的，也就是说，h 一定是可约的，所以 h 可能会泄露一些秘密信息，需要随机化一下；二是 h 是一个 $2t$ 次分享，需要把它降低成 t 次分享才可以。

要解决第一个问题，思路是让各方分享一个 $2t$ 次的 0，所有人把各方的分享加起来，即可得到一个随机的 $2t$ 次分享。要解决第二个问题，思路是将 h 截断，去掉 t 次以上的项。设 $h(x) = \sum_{i\in[2t]} h_i x^i$，令 $\hat{h} = \sum_{i\in[t]} h_i x^i$ 表示 h 去掉 t 次以上项的截断多项式，则 $h(0) = \hat{h}(0) = xy$。剩下的问题就是在各方有 $h(1), \ldots, h(n)$ 的情况下，如何得到 $\hat{h}(1), \ldots, \hat{h}(n)$。实际上，这两组分享只相差一个线性关系，即 $(\hat{h}(1), \ldots, \hat{h}(n))^T = A\cdot(h(1), \ldots, h(n))^T$。下面给出证明：

令 $\vec{h} = (h_0, h_1, \ldots, h_t, \ldots, h_{2t}, 0, \ldots, 0) \in \mathbb{F}^n$，令 V 表示 n 阶范德蒙矩阵，

即 $V = \begin{bmatrix} 1 & 1 & \dots & 1^{n-1} \\ 1 & 2 & \dots & 2^{n-1} \\ \vdots & \vdots & & \vdots \\ 1 & n & \dots & n^{n-1} \end{bmatrix}$

则 $V \cdot \vec{h}^T = (h(1), \dots, h(n))^T$。由于 V 是可逆的，有 $\vec{h}^T = V^{-1} \cdot (h(1), \dots, h(n))^T$

同样地，令 $\vec{\hat{h}} = (h_0, h_1, \dots, h_t, 0, \dots, 0)$，有 $V \cdot \vec{\hat{h}}^T = (\hat{h}(1), \dots, \hat{h}(n))^T$

令 $T = \{1, \dots, t\}$，且令 P_T 表示对角线前 t 个位置为 1，剩下位置为 0 的 n 阶矩阵，即 $P_T(i,j) = 1$ 当且仅当 $i = j \in T$。则 $P_T \cdot \vec{h}^T = \vec{\hat{h}}^T$。于是

$(\hat{h}(1), \dots, \hat{h}(n))^T = V \cdot \vec{\hat{h}}^T = V \cdot P_T \cdot \vec{h}^T = V \cdot P_T \cdot V^{-1} \cdot (h(1), \dots, h(n))^T$

即 $A = V \cdot P_T \cdot V^{-1}$。证完。

有了 $(\hat{h}(1), \dots, \hat{h}(n))^T = A \cdot (h(1), \dots, h(n))^T$ 关系之后，可以发现，$2t$ 次分享和 t 次分享只相差一些线性运算，而线性运算可以在把输入分享之后，直接本地算。乘法就这样做完了。协议描述如下：

1. P_i 计算 $z_i'' := x_i y_i$。
2. P_i 随机选一个 $2t$ 次多项式 $q_i(x)$ 满足 $q_i(0) = 0$，将 $s_{i,j} = q_i(j)$ 发给 $P_j, j \neq i$。
3. P_i 收到 $s_{1,i}, \dots, s_{n,i}$，令 $z_i' := z_i'' + \sum_{j \in [n]} s_{j,i}$。
4. 各方用 z_i' 作为输入，执行 BGW 协议，计算 $A \cdot (z_1', \dots, z_n') = (z_1, \dots, z_n)$，$P_i$ 输出 z_i。则 z_i 就是 xy 的 t 次秘密分享。

这里第二第三步是重随机化，第四步是次数约减。

其实还有另一种角度看待这个乘法协议。如何在次数约减这一步更直观地理解矩阵 A 呢？在 Shamir 分享中，要重构秘密，需要用到拉格朗日插值公式，即过 $(1, x_1), \dots, (n, x_n)$ 的多项式 $f(x)$ 可以写为：

$$f(x) = \sum_{i=1}^{n} x_i \Big(\prod_{\substack{j \neq i}}^{1 \le j \le n} \frac{x - j}{i - j} \Big)$$

有

$$x = f(0) = \sum_{i=1}^{n} x_i \Big(\prod_{\substack{j \neq i}}^{1 \le j \le n} \frac{-j}{i - j} \Big), \text{令} \lambda_i = \prod_{\substack{j \neq i}}^{1 \le j \le n} \frac{-j}{i - j}$$

则 $x = \sum_{i \in [n]} \lambda_i x_i$，因此 Shamir 分享的线性组合可以直接恢复秘密。那么当各方得到 xy 的 $2t$ 次分享之后，各方可以直接把这个分享再用 t 次 Shamir 分享分享出去，然后各方用 λ_i 把收到的分享线性组合，就能得到 xy 的 t 次分享。这样看还有一个好处，就是因为各方使用了新的 t 次分享，此时各方得到的分享一定是随机的，所以重随机步骤可以不要了。重新描述协议如下：

1. P_i 计算 $z'_i := x_i y_i$。
2. P_i 把 z'_i 用 Shamir 分享出去，即随机选取 t 次多项式 $u_i(x)$，满足 $u_i(0) = z'_i$，将 $v_{i,j} = u_i(j)$ 发给 $P_j, j \neq i$。
3. P_i 收到 $v_{1,i}, \ldots, v_{n,i}$，令 $z_i := \lambda_1 v_{1,i} + \cdots + \lambda_n v_{n,i}$。由于 $\{v_{i,j}\}_{j\in[n]}$ 是 z'_i 的 t 次 Shamir 分享，$\{z_i\}_{i\in[n]}$ 就是 $\sum_{i\in[n]} \lambda_i z'_i = z = xy$ 的 t 次分享。这里 λ_i 是 $2t$ 次插值多项式的拉格朗日系数。

2.4.1 重复秘密分享乘法协议

由于重复秘密分享 RSS 各方拥有的分享大小为指数 (C^t_{n-1})，因此不适用于 n 比较大的情况，一般具体考虑 $n = 3, 4, 5$ 的时候会使用这种秘密分享 [AFL+16, BGIN19]，它的好处就是乘法的具体通信量很低，只需发送一个元素。这里以 $n = 3$ 为例。

先回顾一下 $(3, 1)$-RSS 方案，要分享 x，令 $x = x_1 + x_2 + x_3$，则 P_1 拿 x_2, x_3，P_2 拿 x_1, x_3，P_3 拿 x_1, x_2（即 P_i 拿 x_{i-1} 和 x_{i+1}，下标模 3）。要计算乘法

$$\begin{aligned}
xy &= (x_1 + x_2 + x_3)(y_1 + y_2 + y_3) \\
&= x_1y_1 + x_2y_2 + x_3y_3 + x_1y_2 + x_1y_3 + x_2y_1 + x_2y_3 + x_3y_1 + x_3y_2 \\
&= \sum_{i\in[3]} (x_{i+1}y_{i+1} + x_{i+1}y_{i-1} + x_{i-1}y_{i+1}) := \sum_{i\in[3]} z_{i+1}
\end{aligned}$$

可以看到，P_i 可以直接本地直接算出 xy 的加法分享 z_{i+1}。为了保持 RSS 的形式，P_i 把 z_{i+1} 发给 P_{i-1}，此时 P_i 就有了 z_{i+1}, z_{i-1} 满足是 xy 的一个 RSS。但是这里还有一个问题，就是 z_{i+1} 并不是随机的，它是由输入的一些乘积加起来得到的，也会出现像 BGW 中类似的问题，因此也需要重新随机化一下，即，让各方生成一个 0 的分享 $\alpha_1 + \alpha_2 + \alpha_3 = 0$。然后把这个分享加上去，$z_{i+1} := x_{i+1}y_{i+1} + x_{i+1}y_{i-1} + x_{i-1}y_{i+1} + \alpha_i$，即可。

下面描述如何生成这样的 0 分享。其实很简单，只需要 P_i 随机选一个元素 ρ_i 发给 P_{i+1} 即可，然后 P_i 令 $\alpha_i := \rho_i - \rho_{i-1}$。容易验证，$\alpha_1 + \alpha_2 + \alpha_3 = 0$。为了大量生成这样的随机对，可以利用伪随机秘密分享 PRSS 类似的想法，把 ρ 作为 PRF 的密钥，即令 $\alpha_i := F_{\rho_{i-1}}(id) - F_{\rho_i}(id)$，这样就能本地计算无限多个 0 分享。代价是由无条件安全变成了计算安全。

下面描述 3 方 RSS 乘法协议：

1. P_i 计算 $z_{i+1} := x_{i+1}y_{i+1} + x_{i+1}y_{i-1} + x_{i-1}y_{i+1} + \alpha_i$。发给 P_{i-1}。
2. P_i 从 P_{i+1} 收到 z_{i-1}，令分享为 (z_{i-1}, z_{i+1})。

可以看到，在 setup 生成 0 分享之后，每方只需发送一个元素即可计算乘法。

2.5 Offline 乘法协议

如果在得到输入之前可以预先计算一些相关随机对，各方在得到输入之后就可以很快地计算乘法。目前这方面主要有两种方法：beaver triple 和 double sharing。其中 beaver triple 可以同时适用 GMW 和 BGW 协议，而 double sharing 只适用于 BGW 协议，即 honest majority。

2.5.1 Beaver triple

Beaver triple[Bea91] 就是一个随机的乘法分享对 $[a],[b],[c]$(对 BGW 就是 $\langle a\rangle_t,\langle b\rangle_t,\langle c\rangle_t$)，其中 $c=ab$。当有了 beaver triple 时，在线阶段的乘法就可以通过打开两次秘密完成，不需要额外的 OT 或者 HE。相当于把乘法放到 offline 做了。这主要基于一个观察：

$$xy=(x-a+a)\cdot(y-b+b)=(x-a)(y-b)+a(y-b)+b(x-a)+ab$$

在线阶段直接打开 $\rho=x-a,\sigma=y-b$，则 $[xy]=\rho\sigma+\sigma[a]+\rho[b]+[c]$。即 $[a],[b],[c]$ 的线性计算，可以直接在本地完成。协议描述如下：

1. P_i 计算 x_i-a_i, y_i-b_i 并发给所有人。
2. 各方收到秘密后重构 $\rho:=x-a,\sigma:=y-b$。
3. P_i 计算 $z_i:=\rho\sigma+\sigma a_i+\rho b_i+c_i$。

关于 beaver triple 的生成，其实和乘法协议是类似的，把输入换成了随机选取的即可。这里以两方布尔加法分享的 beaver triple 生成为例，协议如下：

1. P_1 随机选 $a_1\leftarrow\{0,1\}$ 作为输入，和 P_2 执行随机 OT，P_1 得到 $m^2_{a_1}$，P_2 得到 (m^2_0,m^2_1)。
2. P_2 随机选 $a_2\leftarrow\{0,1\}$ 作为输入，和 P_1 执行随机 OT，P_2 得到 $m^1_{a_2}$，P_1 得到 (m^1_0,m^1_1)。
3. P_1 令 $b_1:=m^1_0\oplus m^1_1$，P_2 令 $b_2:=m^2_0\oplus m^2_1$。
4. P_1 令 $c_1:=a_1b_1\oplus m^2_{a_1}\oplus x^1_0$，$P_2$ 令 $c_2:=a_2b_2\oplus m^1_{a_2}\oplus x^2_0$。

容易验证，$c_1\oplus c_2=(a_1\oplus a_2)(b_1\oplus b_2)$。类似的，也可以用 HE 做。

关于 BGW 协议 beaver triple 的生成 $\langle a\rangle_t,\langle b\rangle_t,\langle c\rangle_t$，思路是让各方先生成随机 t 分享 $\langle a\rangle_t,\langle b\rangle_t$，然后执行一次乘法协议算 $\langle c\rangle_t$ 即可。生成随机 t 分享目前主要有两种方法。一种基于伪随机秘密分享 (psudorandom secret sharing,PRSS)[CDI05]，该方法使各方能够生成伪随机元素的分享，而无需任何交互。主要使用的就是之前 RSS 转化成 Shamir 分享的思路。这种方法的问题是，每一方持有的密钥数量随着参与方的数量呈指数增长，这大大增加了生成分享的计算工作量。因此，只有当协议中的参与方数量较少时，这种方法才有效。具体协议描述如下：

1. (Setup) 对每个 $A \subset [n], |A| = t$，对不在集合 A 中最小的那一方 (即 $P_i, i = min\{j \in [n] | j \notin A\}$) 随机选择 k_A，并发给其他所有不在 A 中的参与方 (即 $P_j, j \notin A$)。最后 P_i 就会得到所有 $\{k_A | i \notin A\}$。

2. (Upon request) 各方对每个 $A \subset [n], |A| = t$ 构造 t 次多项式 f_A，满足 $f_A(0) = 1$,$f_A(i) = 0$ for $i \in A$。P_i 令 $r_i = \sum_{A \subset [n]:|A|=t, i \notin A} F_{k_A}(id) \cdot f_A(i)$。这里 id 是公开参数，每生成一次新的分享都要用一个新的 id。

一种是基于范德蒙矩阵 [DN07]，该矩阵可用于从 n 个分享“提取随机性”到 $t+1$ 个新分享。思路是是让每一方向其他方分享一个随机元素。然后，在持有 n 个分享向量时，各方将该大小为 n 的向量与 $n-t$ 行 n 列的范德蒙矩阵相乘，得到 $n-t$ 个“新”随机分享。通过随机性提取性质，我们得到新的分享是 $\mathbb{F}$ 中随机元素的分享。由于 $t < \frac{n}{2}$，各方得到至少 $\frac{n}{2}+1$ 个分享。由于每方需要发送 $n-1$ 个元素，平均每个随机分享通信约为 2 个元素。设 $V_{n-t} = \begin{bmatrix} 1 & 1 & \dots & 1^{n-t-1} \\ 1 & 2 & \dots & 2^{n-t-1} \\ \vdots & \vdots & & \vdots \\ 1 & n & \dots & n^{n-t-1} \end{bmatrix}$ 是一个 $n \times (n-t)$ 的范德蒙矩阵，协议描述如下：

1. P_i 随机选一个 u_i 并用 t 次 Shamir 分享把 u_i 分享出去，即 $\langle u_i \rangle_t$。

2. 各方本地计算 $(\langle r_1 \rangle_t, \dots, \langle r_{n-t} \rangle_t)^T := V_{n-t}^T \cdot (\langle u_1 \rangle_t, \dots, \langle u_n \rangle_t)^T$。

这里我的理解是这样的，一般考虑生成一个随机分享，平凡的想法是让各方都随机分享一个值，最后把这 n 个值加起来。但是这样效率太慢了，n 个人每个人分享一次才能得到一个随机分享。而又不能直接用每个人的分享，因为分享者知道秘密，所以就要考虑用一个随机提取器，从各方分享的秘密中尽可能多的提取出随机分享。那么能提取出多少随机分享呢，因为有 t 个坏人，所以可以认为坏人提供的分享不具有随机性，或者具有坏的随机性，那么好人的随机性就是 $n-t$ 个。如何从这些好坏掺杂的分享中提取出真正的随机性呢？思路就是把所有分享乘固定系数加起来在一起，把好人的分享加到坏人的分享上，这样就能保证得到的新分享是之前分享的线性组合，坏人的坏随机性被好人真正的随机性吸收掉了 (如 a 是随机的，则对任何 b 的分布，$a+b$ 也是随机的)。而不同的固定系数需要线性无关，才能保证生成的不同的分享也是独立的。考虑到好人提供的随机性至多 $n-t$ 个，因此最多生成 $n-t$ 个新的随机秘密分享。

2.5.2 Double sharing

Double sharing[DN07] 是指同一个秘密的 t 次和 $2t$ 次分享，即 $(\langle r \rangle_t, \langle r \rangle_{2t})$。可以看到这种形式是专门针对 honest majority 的。使用 double sharing 的好处是可以将乘法协议的通信复杂度降到 $O(n)$。注意到之前的协议，不管是 GMW 还是 BGW，还是 beaver triple，都需要每次乘法每个人向其他所有人发送消息，即总的通信复杂度为 $O(n^2)$。

使用 double sharing 的主要思想是选取某一方作为 P_{king}，让大家先把自己的分享本地相乘，再减去 $2t$ 次随机分享 r，即 $\langle x\rangle_t\cdot\langle y\rangle_t-\langle r\rangle_{2t}$，发给 P_{king}。P_{king} 重构出 $xy-r$ 之后再发给各方，各方收到之后在加上 r 的 t 次分享，即 $xy-r+\langle r\rangle_t$ 这样就得到了 xy 的 t 次分享 $\langle xy\rangle_t$。设 $\langle r\rangle_t$ 和 $\langle r\rangle_{2t}$ 中各方的分享分别是 r_i^t 和 r_i^{2t}。使用 double sharing 的乘法协议如下：

1. P_i 计算 $x_iy_i-r_i^{2t}$ 并发给 P_{king}。
2. P_{king} 重构出 $xy-r$，发给所有人。
3. P_i 令 $z_i:=xy-r+r_i^t$。

double sharing 的生成和单个随机 BGW 分享 $\langle r\rangle_t$ 的生成类似，也是使用范德蒙矩阵进行提取。这里不能使用 PRSS 的方法是因为我们需要同时生成同一个数的 t 和 $2t$ 的分享，而 PRSS 中的门限和 RSS 的门限是一样的，这意味着要想生成 $2t$ 次分享，还要再进行一个 $2t$ 的 setup，但是即使 setup 了之后，由于每次运行都是生成新的伪随机数的分享，无法保证和 t 门限时候的值一致。这里描述用范德蒙矩阵生成 double sharing 的协议如下：

1. P_i 随机选一个 u_i 并同时用 t 次和 $2t$ 次 Shamir 分享把 u_i 分享出去，即 $\langle u_i\rangle_t,\langle u_i\rangle_{2t}$。
2. 各方本地计算
   $(\langle r_1\rangle_t,\dots,\langle r_{t+1}\rangle_t)^T:=V_{t+1}^T\cdot(\langle u_1\rangle_t,\dots,\langle u_n\rangle_t)^T$,
   $(\langle r_1\rangle_{2t},\dots,\langle r_{t+1}\rangle_{2t})^T:=V_{t+1}^T\cdot(\langle u_1\rangle_{2t},\dots,\langle u_n\rangle_{2t})^T$

2.6 GC 乘法协议

GC 其实没有乘法协议，倒不如说，GC 的生成本身就是一个非交互乘法协议，采用点置换技术，置换比特实际上也起到了一个秘密分享的作用。对门 (a,b,c) 来说，设真值分别是 (z_a,z_b,z_c)，garbler 拥有 $(\lambda_a,\lambda_b,\lambda_c)$，而 evaluator 拥有 $(\hat{z_a},\hat{z_b},\hat{z_c})=(z_a\oplus\lambda_a,z_b\oplus\lambda_b,z_c\oplus\lambda_c)$，恰好是一个加法分享。

接下来 garbler 要做的就是如何让 evaluator 求值的时候得到对应的输出分享，我们来看一下现在双方都有什么。对乘法门 (a,b,c)，garbler 有输入线的分享 λ_a,λ_b 和输出线的分享 λ_c，evaluator 有输入线的分享 $\hat{z_a}=z_a\oplus\lambda_a,\hat{z_b}=z_b\oplus\lambda_b$，目标是让 evaluator 得到输出线的分享 $\hat{z_c}=z_c\oplus\lambda_c$

这里其实用的就是一个 GMW 协议中 4 选 1OT 做乘法的思路，garbled table 的四行密文其实恰恰对应了 OT 的四条消息，能够让 evaluator 求出对应的输出。

回顾点置换和 FreeXOR 技术的 garbled table：

$e_{0,0}=H(L_{a,0}||L_{b,0}||g)\oplus L_{c,0}\oplus(\lambda_a\cdot\lambda_b\oplus\lambda_c)\Delta$

$e_{0,1}=H(L_{a,0}||L_{b,1}||g)\oplus L_{c,0}\oplus(\lambda_a\cdot\overline{\lambda_b}\oplus\lambda_c)\Delta$

$e_{1,0}=H(L_{a,1}||L_{b,0}||g)\oplus L_{c,0}\oplus(\overline{\lambda_a}\cdot\lambda_b\oplus\lambda_c)\Delta$

$e_{1,1} = H(L_{a,1}||L_{b,1}||g) \oplus L_{c,0} \oplus (\overline{\lambda_a} \cdot \overline{\lambda_b} \oplus \lambda_c)\Delta$

如果我们用 GMW 中的符号，$x_1 = \lambda_a, y_1 = \lambda_b, x_2 = \hat{z_z}, y_2 = \hat{z_b}, z_1 = \lambda_c$，这时候可以发现，四条密文对应的恰好是 4 选 1OT 里的四条消息。由于 evaluator 只有一组正确的 label 对，因此只能解密一行，就是对应 $z_2 = \hat{z_c}$ 那一行。满足 $z_1 \oplus z_2 = (x_1 \oplus x_2)(y_1 \oplus y_2)$。

个人的理解：从这个角度看，GC 和 GMW 做乘法的想法是类似的，但是为什么 GC 可以做到常数轮，而 GMW 的轮数和乘法深度有关呢？我认为是 GC 是做了一个轮数和通信的 trade-off，即通过大量增加通信换取了常数轮数。在 GMW 中，每次做乘法其实只需要把真值对应的那一条消息发过去就行了，使用 OT extension 技术，用 OT 传一个比特平均通信就是 $O(1)$。而 GC 则需要一次性把所有可能的消息遍历一遍都发过去。而为了防止泄露额外信息，那些非真值的消息要用加密掩盖，而这就导致密文的长度至少是安全参数 $O(\kappa)$，即传一个比特要通信 $O(\kappa)$。

参考文献

[AFL+16] Toshinori Araki, Jun Furukawa, Yehuda Lindell, Ariel Nof, and Kazuma Ohara. High-throughput semi-honest secure three-party computation with an honest majority. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 805–817, 2016.

[AIK11] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. How to garble arithmetic circuits. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 120–129, 2011.

[AL07] Yonatan Aumann and Yehuda Lindell. Security against covert adversaries: Efficient protocols for realistic adversaries. In *Theory of Cryptography, 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007, Proceedings*, pages 137–156, 2007.

[AL17] Gilad Asharov and Yehuda Lindell. A full proof of the BGW protocol for perfectly secure multiparty computation. *J. Cryptology*, 30(1):58–151, 2017.

[AO12] Gilad Asharov and Claudio Orlandi. Calling out cheaters: Covert security with public verifiability. In *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, pages 681–698, 2012.

[BCM+19] Marshall Ball, Brent Carmer, Tal Malkin, Mike Rosulek, and Nichole Schimanski. Garbled neural networks are practical. *IACR Cryptol. ePrint Arch.*, 2019:338, 2019.

[BDOZ11] Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. Semi-homomorphic encryption and multiparty computation. In *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, pages 169–188, 2011.

[Bea91] Donald Beaver. Efficient multiparty protocols using circuit randomization. In *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, pages 420–432, 1991.

[Ben18] Aner Ben-Efraim. On multiparty garbling of arithmetic circuits. In *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part III*, pages 3–33, 2018.

[BFO12] Eli Ben-Sasson, Serge Fehr, and Rafail Ostrovsky. Near-linear unconditionally-secure multiparty computation with a dishonest minority. In *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, pages 663–680, 2012.

[BGIN19] Elette Boyle, Niv Gilboa, Yuval Ishai, and Ariel Nof. Practical fully secure three-party computation via sublinear distributed zero-knowledge proofs. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, pages 869–886, 2019.

[BGIN20] Elette Boyle, Niv Gilboa, Yuval Ishai, and Ariel Nof. Efficient fully secure computation via distributed zero-knowledge proofs. *IACR Cryptol. ePrint Arch.*, 2020:1451, 2020.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 1–10, 1988.

[BH92] Donald Beaver and Stuart Haber. Cryptographic protocols provably secure against dynamic adversaries. In *Advances in Cryptology - EUROCRYPT '92, Workshop on the Theory and Application of of Cryptographic Techniques, Balatonfüred, Hungary, May 24-28, 1992, Proceedings*, pages 307–323, 1992.

[BHR12] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In *the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012*, pages 784–796, 2012.

[BLN+15] Sai Sheshank Burra, Enrique Larraia, Jesper Buus Nielsen, Peter Sebastian Nordholt, Claudio Orlandi, Emmanuela Orsini, Peter Scholl, and Nigel P. Smart. High performance multi-party computation for binary circuits based on oblivious transfer. *IACR Cryptology ePrint Archive*, 2015:472, 2015.

[BLO16] Aner Ben-Efraim, Yehuda Lindell, and Eran Omri. Optimizing semi-honest secure multiparty computation for the internet. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 578–590, 2016.

[BLO17] Aner Ben-Efraim, Yehuda Lindell, and Eran Omri. Efficient scalable constant-round MPC via garbled circuits. In *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part II*, pages 471–498, 2017.

[BMR90] Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 503–513, 1990.

[BMR16] Marshall Ball, Tal Malkin, and Mike Rosulek. Garbling gadgets for boolean and arithmetic circuits. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 565–577, 2016.

[Can00] Ran Canetti. Security and composition of multiparty cryptographic protocols. *J. Cryptol.*, 13(1):143–202, 2000.

[CDE+18] Ronald Cramer, Ivan Damgård, Daniel Escudero, Peter Scholl, and Chaoping Xing. Spd$\mathcal{F}_{2^k}$: Efficient MPC mod 2^k for dishonest majority. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part II*, pages 769–798, 2018.

[CDI05] Ronald Cramer, Ivan Damgård, and Yuval Ishai. Share conversion, pseudorandom secret-sharing and applications to secure computation. In *Theory of Cryptography,* *Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings*, pages 342–362, 2005.

[CDM00] Ronald Cramer, Ivan Damgård, and Ueli M. Maurer. General secure multi-party computation from any linear secret-sharing scheme. In *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding*, pages 316–334, 2000.

[CFGN96] Ran Canetti, Uriel Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 639–648, 1996.

[CGH+18] Koji Chida, Daniel Genkin, Koki Hamada, Dai Ikarashi, Ryo Kikuchi, Yehuda Lindell, and Ariel Nof. Fast large-scale honest-majority MPC for malicious adversaries. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part III*, pages 34–64, 2018.

[CGZ20] Ran Cohen, Juan A. Garay, and Vassilis Zikas. Broadcast-optimal two-round MPC. In *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part II*, pages 828–858, 2020.

[CH94] Ran Canetti and Amir Herzberg. Maintaining security in the presence of transient faults. In *Advances in Cryptology - CRYPTO '94, 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1994, Proceedings*, pages 425–438, 1994.

[Cle86] Richard Cleve. Limits on the security of coin flips when half the processors are faulty (extended abstract). In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*, pages 364–369, 1986.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 494–503, 2002.

[DKL+13] Ivan Damgård, Marcel Keller, Enrique Larraia, Valerio Pastro, Peter Scholl, and Nigel P. Smart. Practical covertly secure MPC for dishonest majority - or: Breaking the SPDZ limits. In *Computer Security - ESORICS 2013 - 18th European Symposium on Research in Computer Security, Egham, UK, September 9-13, 2013. Proceedings*, pages 1–18, 2013.

[DN07] Ivan Damgård and Jesper Buus Nielsen. Scalable and unconditionally secure multiparty computation. In *Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings*, pages 572–590, 2007.

[DOS20] Ivan Damgård, Claudio Orlandi, and Mark Simkin. Black-box transformations from passive to covert security with public verifiability. In *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part II*, pages 647–676, 2020.

[DPSZ12] Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, pages 643–662, 2012.

[EKR18] David Evans, Vladimir Kolesnikov, and Mike Rosulek. A pragmatic introduction to secure multi-party computation. *Foundations and Trends in Privacy and Security*, 2(2-3):70–246, 2018.

[FHKS21] Sebastian Faust, Carmit Hazay, David Kretzler, and Benjamin Schlosser. Generic compiler for publicly verifiable covert multi-party computation. In *Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part II*, pages 782–811, 2021.

[FL19] Jun Furukawa and Yehuda Lindell. Two-thirds honest-majority MPC for malicious adversaries at almost the cost of semi-honest. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, pages 1557–1571, 2019.

[FLNW17] Jun Furukawa, Yehuda Lindell, Ariel Nof, and Or Weinstein. High-throughput secure three-party computation for malicious adversaries and an honest majority. In *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International*

Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part II, pages 225–255, 2017.

[GIP+14] Daniel Genkin, Yuval Ishai, Manoj Prabhakaran, Amit Sahai, and Eran Tromer. Circuits resilient to additive attacks with applications to secure computation. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 495–504, 2014.

[GLO+21] Vipul Goyal, Hanjun Li, Rafail Ostrovsky, Antigoni Polychroniadou, and Yifan Song. ATLAS: efficient and scalable MPC in the honest majority setting. In *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part II*, pages 244–274, 2021.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 218–229, 1987.

[Gol04] Oded Goldreich. *The Foundations of Cryptography - Volume 2: Basic Applications*. Cambridge University Press, 2004.

[GS20] Vipul Goyal and Yifan Song. Malicious security comes free in honest-majority MPC. *IACR Cryptol. ePrint Arch.*, 2020:134, 2020.

[GSZ20] Vipul Goyal, Yifan Song, and Chenzhi Zhu. Guaranteed output delivery comes free in honest majority MPC. In *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part II*, pages 618–646, 2020.

[HKK+19] Cheng Hong, Jonathan Katz, Vladimir Kolesnikov, Wen-jie Lu, and Xiao Wang. Covert security with public verifiability: Faster, leaner, and simpler. In *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part III*, pages 97–121, 2019.

[HSS17] Carmit Hazay, Peter Scholl, and Eduardo Soria-Vazquez. Low cost constant round MPC combining BMR and oblivious transfer. In *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I*, pages 598–628, 2017.

[Kel20] Marcel Keller. MP-SPDZ: A versatile framework for multi-party computation. In *CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020*, pages 1575–1590, 2020.

[KM15] Vladimir Kolesnikov and Alex J. Malozemoff. Public verifiability in the covert model (almost) for free. In *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II*, pages 210–235, 2015.

[KOS16] Marcel Keller, Emmanuela Orsini, and Peter Scholl. MASCOT: faster malicious arithmetic secure computation with oblivious transfer. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 830–842, 2016.

[KPR18] Marcel Keller, Valerio Pastro, and Dragos Rotaru. Overdrive: Making SPDZ great again. In *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part III*, pages 158–189, 2018.

[KRRW18] Jonathan Katz, Samuel Ranellucci, Mike Rosulek, and Xiao Wang. Optimizing authenticated garbling for faster secure two-party computation. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part III*, pages 365–391, 2018.

[KS08] Vladimir Kolesnikov and Thomas Schneider. Improved garbled circuit: Free XOR gates and applications. In *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II - Track B: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography Foundations*, pages 486–498, 2008.

[Lin09] Andrew Y. Lindell. Adaptively secure two-party computation with erasures. In *Topics in Cryptology - CT-RSA 2009, The Cryptographers' Track at the RSA Conference 2009, San Francisco, CA, USA, April 20-24, 2009. Proceedings*, pages 117–132, 2009.

[Lin13] Yehuda Lindell. Fast cut-and-choose based protocols for malicious and covert adversaries. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology*

Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II, pages 1–17, 2013.

[LN17] Yehuda Lindell and Ariel Nof. A framework for constructing fast MPC over arithmetic circuits with malicious adversaries and an honest-majority. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 259–276, 2017.

[LP04] Yehuda Lindell and Benny Pinkas. A proof of yao's protocol for secure two-party computation. *Electronic Colloquium on Computational Complexity (ECCC)*, (063), 2004.

[LP07] Yehuda Lindell and Benny Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. In *Advances in Cryptology - EUROCRYPT 2007, 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Barcelona, Spain, May 20-24, 2007, Proceedings*, pages 52–78, 2007.

[LP11] Yehuda Lindell and Benny Pinkas. Secure two-party computation via cut-and-choose oblivious transfer. In *Theory of Cryptography - 8th Theory of Cryptography Conference, TCC 2011, Providence, RI, USA, March 28-30, 2011. Proceedings*, pages 329–346, 2011.

[LPSY15] Yehuda Lindell, Benny Pinkas, Nigel P. Smart, and Avishay Yanai. Efficient constant round multi-party computation combining BMR and SPDZ. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, pages 319–338, 2015.

[LR14] Yehuda Lindell and Ben Riva. Cut-and-choose yao-based secure computation in the online/offline and batch settings. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II*, pages 476–494, 2014.

[LR15] Yehuda Lindell and Ben Riva. Blazing fast 2pc in the offline/online setting with security for malicious adversaries. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, pages 579–590, 2015.

[LSS16] Yehuda Lindell, Nigel P. Smart, and Eduardo Soria-Vazquez. More efficient constant-round multi-party computation from BMR and SHE. In *Theory of Cryp-*

tography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part I, pages 554–581, 2016.

[MW19] Eleftheria Makri and Tim Wood. Full-threshold actively-secure multiparty arithmetic circuit garbling. *IACR Cryptol. ePrint Arch.*, 2019:1098, 2019.

[NNOB12] Jesper Buus Nielsen, Peter Sebastian Nordholt, Claudio Orlandi, and Sai Sheshank Burra. A new approach to practical active-secure two-party computation. In *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, pages 681–700, 2012.

[NPS99] Moni Naor, Benny Pinkas, and Reuban Sumner. Privacy preserving auctions and mechanism design. In *Proceedings of the First ACM Conference on Electronic Commerce (EC-99), Denver, CO, USA, November 3-5, 1999*, pages 129–139, 1999.

[NV18] Peter Sebastian Nordholt and Meilof Veeningen. Minimising communication in honest-majority MPC by batchwise multiplication verification. In *Applied Cryptography and Network Security - 16th International Conference, ACNS 2018, Leuven, Belgium, July 2-4, 2018, Proceedings*, pages 321–339, 2018.

[OY91] Rafail Ostrovsky and Moti Yung. How to withstand mobile virus attacks (extended abstract). In *Proceedings of the Tenth Annual ACM Symposium on Principles of Distributed Computing, Montreal, Quebec, Canada, August 19-21, 1991*, pages 51–59, 1991.

[PSSW09] Benny Pinkas, Thomas Schneider, Nigel P. Smart, and Stephen C. Williams. Secure two-party computation is practical. In *Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings*, pages 250–267, 2009.

[RR16] Peter Rindal and Mike Rosulek. Faster malicious 2-party secure computation with online/offline dual execution. In *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016.*, pages 297–314, 2016.

[RR21] Mike Rosulek and Lawrence Roy. Three halves make a whole? beating the half-gates lower bound for garbled circuits. In *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part I*, pages 94–124, 2021.

[SSS21] Peter Scholl, Mark Simkin, and Luisa Siniscalchi. Multiparty computation with covert security and public verifiability. *IACR Cryptol. ePrint Arch.*, page 366, 2021.

[WMK17] Xiao Wang, Alex J. Malozemoff, and Jonathan Katz. Faster secure two-party computation in the single-execution setting. In *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part III*, pages 399–424, 2017.

[WRK17a] Xiao Wang, Samuel Ranellucci, and Jonathan Katz. Authenticated garbling and efficient maliciously secure two-party computation. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 21–37, 2017.

[WRK17b] Xiao Wang, Samuel Ranellucci, and Jonathan Katz. Global-scale secure multiparty computation. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 39–56, 2017.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pages 162–167, 1986.

[YWZ20] Kang Yang, Xiao Wang, and Jiang Zhang. More efficient MPC from improved triple generation and authenticated garbling. In *CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020*, pages 1627–1646, 2020.

[ZRE15] Samee Zahur, Mike Rosulek, and David Evans. Two halves make a whole - reducing data transfer in garbled circuits using half gates. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*, pages 220–250, 2015.