通用 MPC 协议构造

常数轮 MPC 协议：BMR 框架

张聪
zhangcong@iie.ac.cn

中国科学院信息工程研究所国家重点实验室

2023 年 5 月 6 日

1. 介绍
2. BMR 框架
3. 参考文献

BMR 介绍

由于混淆电路协议只适用于两方，Beaver,Micali 和 Rogaway[BMR90] 提出了 BMR 协议框架，将 GC 推广到了多方，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。

BMR 介绍

目前 BMR 框架大概有如下发展:

- 半诚实情况：最早的 BMR 协议 [BMR90] 只是说使用 MPC 协议 (例如 GMW, BGW) 进行分布式生成 GC，为了避免在 MPC 内部求值 PRF 电路，让各方本地计算 PRF 值，用各方的 PRF 的异或值作为加密密钥；[BLO16] 将 FreeXOR 技术应用到 BMR 框架中，并基于 OT 给出了具体的分布式生成 GC 的协议；[BLO17] 使用密钥同态 PRF 将协议在线阶段的通信复杂度从 $O(n^2)$ 降到 $O(1)$，在参与方人数很多 (> 100) 时效率更高，但此技术不与 FreeXOR 兼容；[Ben+21] 则是基于 LPN 假设构造了同时满足消息同态和密钥同态的 PRF，使得之前的协议满足低通信的同时与 FreeXOR 兼容。
- 恶意情况：[Lin+15; LSS16] 分别使用 SPDZ 和 SHE 对每个 garbled table 做认证，[HSS17; WRK17; 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。

BMR 协议

主要思路：为了计算电路 f，各方首先运行一个 MPC 协议 (GMW/BGW) 计算一个生成电路 f 的 GC 的电路。设 C_{gen}^{f} 是生成电路 f 的 GC 的电路，由于对于任意电路，每个门的 garbled table 可以并行生成，因此 C_{gen}^{f} 的深度和具体的电路 f 的深度无关，是一个确定的常数。当各方使用 GMW/BGW 协议计算 C_{gen}^{f} 时，通信轮数是常数。生成 GC 之后，指定一个参与方 (如 P_1) 进行求值，或者让所有参与方都进行求值，得到输出。

BMR 协议

问题 1：回忆 garbled table 构造，$G = F(L_a, L_b) \oplus L_c$，此时需要在 MPC 协议内部求值 PRF 电路，如何避免？

BMR 协议

问题 1：回忆 garbled table 构造，$G = F(L_a, L_b) \oplus L_c$，此时需要在 MPC 协议内部求值 PRF 电路，如何避免？

解决思路：让每个参与方共同生成 label 的一部分，对门 (a,b,c,g) 令 garbled table 为 $G_{z_a,z_b} := L_{c,z_c} \oplus (\bigoplus_{i\in[n]} F(L^i_{a,z_a}, c) \oplus F(L^i_{b,z_b}, c))$
这里 $F : \{0,1\}^\kappa \to \{0,1\}^{n\kappa}$ 是 PRG，每个参与方 P_i 为每条线 w 选两个 label $L^i_{w,0}, L_{w,1} \in \{0,1\}^\kappa$，真实的 label 是所有参与方的 label 的并 $L_w := L^1_w || \ldots || L^n_w \in \{0,1\}^{n\kappa}$
此时每个参与方可以本地算好 PRG 的值作为 MPC 协议的输入，MPC 协议内部只需把所有值加起来，不需要求值 PRF 电路。

BMR 协议

问题 2：在这种情况下，P_1 求值时会得到输入线上的 label (e.g. $L_{w,x_w} = L^1_{w,x_w} || \dots || L^n_{w,x_w}$)，但此时，由于 P_1 也参与了 GC 的构造，对这个 label，他是知道 $L^1_{w,0}$ 和 $L^1_{w,1}$，因此只需对比一下对应位置的 label，就能知道这条线上的真值 x_w 是多少。

BMR 协议

问题 2：在这种情况下，P_1 求值时会得到输入线上的 label (e.g. $L_{w,x_w} = L^1_{w,x_w} || \dots || L^n_{w,x_w}$)，但此时，由于 P_1 也参与了 GC 的构造，对这个 label，他是知道 $L^1_{w,0}$ 和 $L^1_{w,1}$，因此只需对比一下对应位置的 label，就能知道这条线上的真值 x_w 是多少。

解决思路：让每个参与方 P_i 为每条线 w 再选择一个翻转比特 (flip bit) f^i_w，令 $f_w := \bigoplus_{i \in [n]} f^i_w$ 表示是否把 label 表示真值的含义翻转，此时 $L_{w,0}$ 不再表示真值 0，而是 f_w。这样，即使 P_1 拿到 label，也无法通过对比知道真值，因为翻转比特是所有参与方共同生成的。

BMR 协议

设电路 C 线的集合是 W，定义理想功能 $\mathcal{F}^C_{Gen}$：

1. 从每个参与方 P_i 接受输入
$I^i = \{I^i_w\}_{w\in W} = \{F(L^i_{w,0}), F(L^i_{w,1}), L^i_{w,0}, L^i_{w,1}, \lambda^i_w, f^i_w\}_{w\in W}$
2. 对每个门 (a,b,c,g)，计算 $\lambda_a := \bigoplus_{i\in[n]} \lambda^i_a$，$\lambda_b := \bigoplus_{i\in[n]} \lambda^i_b$，$\lambda_c := \bigoplus_{i\in[n]} \lambda^i_c$，
$f_a := \bigoplus_{i\in[n]} f^i_a$，$f_b := \bigoplus_{i\in[n]} f^i_b$，$f_c := \bigoplus_{i\in[n]} f^i_c$
3. 对每个门 (a,b,c,g)，对所有真值 $z_a, z_b \in \{0,1\}$ 计算 garbled table：
$$G_{z_a,z_b} := (\bigoplus_{i\in[n]} F(L^i_{a,z_a\oplus f_a}, c) \oplus F(L^i_{b,z_b\oplus f_b}, c)) \oplus L_{c,z_c\oplus f_c}$$
这里 $L_{c,0} = L^1_{c,0}||\ldots||L^n_{c,0}||\lambda_c, L_{c,1} = L^1_{c,1}||\ldots||L^n_{c,1}||\bar{\lambda}_c$
将 G_{z_a,z_b} 放在第 $(\lambda_a \oplus z_a, \lambda_b \oplus z_b)$ 行
4. 令输出解密表 $d := \{(0, L_{w,f_w}), (1, L_{w,\bar{f}_w})\}_{w\in Outputs}$
5. 将所有输入线 label $\{L_{w,x_w\oplus f_w}\}_{w\in Inputs}$，输出解密表 d 和所有门的 garbled table $\vec{G}$ 发给 P_1

BMR 协议

设 $F:\{0,1\}^\kappa \to \{0,1\}^{n\kappa}$ 是 PRG，BMR 协议描述如下：

1. 对电路 C 的每条线 $w \in W$，参与方 P_i 随机选 $L^i_{w,0}, L^i_{w,1} \leftarrow \{0,1\}^\kappa$ 分别表示对应真值 $0,1$ 的 label，令置换比特 $\lambda^i_w := lsb(L^i_{w,0})$，再选择翻转比特 $f^i_w \leftarrow \{0,1\}$。令 $I^i = \{I^i_w\}_{w\in W} = \{F(L^i_{w,0}), F(L^i_{w,1}), L^i_{w,0}, L^i_{w,1}, \lambda^i_w, f^i_w\}_{w\in W}$
2. 每个参与方 P_i 用输入 I^i 调用 $\mathcal{F}^C_{Gen}$，P_1 得到输出 $\{L_{w,x_w\oplus f_w}\}_{w\in Inputs}, \vec{G}, d$
3. P_1 求值 GC，得到输出 $C(x_1,\ldots,x_n)$ 并发给其他参与方

BMR 协议

上述协议来自 [EKR18]，但是我发现一个问题，这个构造可能是不对的。
这里他们给的 garbled table 构造公式为：

$$G_{z_a,z_b} := (\bigoplus_{i\in[n]} F(L^i_{a,z_a\oplus f_a}, c) \oplus F(L^i_{b,z_b\oplus f_b}, c)) \oplus L_{c,z_c\oplus f_c}$$

这一行代表的真值是 (z_a, z_b)，并把这一个密文根据置换比特放在了 $(\lambda_a \oplus z_a, \lambda_b \oplus z_b)$ 行。也就是说，第 $(\lambda_a \oplus z_a, \lambda_b \oplus z_b)$ 行加密的真值是 (z_a, z_b)。
换句话说，garbled table 的第 (u, v) 行表示的真值是 $(u \oplus \lambda_a, v \oplus \lambda_b)$。用的加密 label 是 $L_{a,u\oplus\lambda_a\oplus f_a}, L_{b,v\oplus\lambda_b\oplus f_b}$
再回忆一下前面的说明，由于担心求值者 P_1 根据 L_{a,z_a} 和 L^1_{a,z_a} 对比得到这条线上的真值 z_a，因此每个人选择一个翻转比特，使得 L_{a,z_a} 表示的真值不再是 z_a，而是 $z_a \oplus f_a$。换句话说，$L_{a,z_a\oplus f_a}$ 表示的真值才是 z_a。
再看协议里的定义，有 $lsb(L_{a,0}) = \lambda_a$，则可以得到 $lsb(L_{a,z_a\oplus f_a}) = \lambda_a \oplus z_a \oplus f_a$，即表示真值 z_a 的 label 的最后一比特是 $\lambda_a \oplus z_a \oplus f_a$

BMR 协议

有了以上符号表示之后，下面来看如何求值这个 GC：
首先明确一点，GC 的求值思路是按照电路门的拓扑排序逐个门求值，主要思路是根据输入线上的 active label 求得输出线上的 active label。
当 P_1 求值这个 GC 时，首先拿到所有输入线上的 active label，也就是代表真值的 label。对于门 (a, b, c, G) 来说，就是 $L_{a,z_a \oplus f_a}, L_{b,z_b \oplus f_b}$，他需要求得 $L_{c,z_c \oplus f_c}$。那么拿到输入 label 之后，他首先需要根据置换比特找到对应置换比特的那一行密文进行解密，那按照 point-and-permute GC 的思路，首先应该求 $\hat{a} := lsb(L_{a,z_a \oplus f_a}) = \lambda_a \oplus z_a \oplus f_a, \hat{b} := lsb(L_{b,v_b \oplus f_b}) = \lambda_b \oplus z_b \oplus f_b$，然后解密 garbled table 的 $(\hat{a}, \hat{b})$ 这一行的密文。根据前面所述，这一行的密文加密的真值是 $(\hat{a} \oplus \lambda_a, \hat{b} \oplus \lambda_b) = (z_a \oplus f_a, z_b \oplus f_b)$，用的加密 label 是 $L_{a,z_a \oplus f_a \oplus f_a} = L_{a,z_a}, L_{b,z_b \oplus f_b \oplus f_b} = L_{b,z_b}$。这显然和他求值用的 label(即 $L_{a,z_a \oplus f_a}, L_{b,z_b \oplus f_b}$) 不同，因此不可能求出正确的输出 label。

BMR 协议

按我理解，错误出现的原因是，这里的求值行数出错了，不应该求 $(\hat{a}, \hat{b}) = (\lambda_a \oplus z_a \oplus f_a, \lambda_b \oplus z_b \oplus f_b)$ 这一行，而是应该求 $(\lambda_a \oplus z_a, \lambda_b \oplus z_b)$ 这一行才对。但是这两个值不包含在输入 label 中，无法让求值者得到。但是就算让求值者得到正确的解密行数值 (即 $(\lambda_a \oplus z_a, \lambda_b \oplus z_b)$)，这个方案也就不安全了，因为再结合 label 的 lsb 给的信息，求值者完全可以恢复 f_a, f_b 的信息，这时候 f_a, f_b 就完全失去了他们原本的作用。

综上所述，似乎置换比特和翻转比特合二为一才是安全的方案，不能像现在写的这样，分开用两个比特表示。此时，P_i 还是对每条线 w 选择两个 label, $L^i_{w,0}, L^i_{w,1}$，但此时下标的 0, 1 不再表示真值，表示哪个真值要根据所有人选取的置换比特决定。每个人再选一个置换比特 λ^i_w。令 $\lambda_w := \oplus_{i \in [n]} \lambda^i_w$, $L_{w,\lambda_w} := L^1_{w,\lambda_w} || \dots || L^n_{w,\lambda_w} || \lambda_w$, $L_{w,\bar{\lambda}_w} := L^1_{w,\bar{\lambda}_w} || \dots || L^n_{w,\bar{\lambda}_w} || \bar{\lambda}_w$，这里 L_{w,λ_w} 表示 0 的 label，$L_{w,\bar{\lambda}_w}$ 表示 1 的 label。注意，此时 $lsb(L_{w,0}) = 0, lsb(L_{w,1}) = 1$，下标即表示最后一个比特而不是真值。

BMR 协议

设电路 C 线的集合是 W，定义理想功能 $\mathcal{F}_{Gen}^{C}$：

1. 从每个参与方 P_i 接受输入 $I^i = \{I_w^i\}_{w\in W} = \{F(L_{w,0}^i), F(L_{w,1}^i), L_{w,0}^i, L_{w,1}^i, \lambda_w^i\}_{w\in W}$
2. 对每个门 (a,b,c,g)，计算 $\lambda_a := \bigoplus_{i\in[n]} \lambda_a^i$，$\lambda_b := \bigoplus_{i\in[n]} \lambda_b^i$，$\lambda_c := \bigoplus_{i\in[n]} \lambda_c^i$
3. 对每个门 (a,b,c,g)，对所有真值 $z_a, z_b \in \{0,1\}$ 计算 garbled table：
   $$G_{z_a,z_b} := (\bigoplus_{i\in[n]} F(L_{a,z_a\oplus\lambda_a}^i, c) \oplus F(L_{b,z_b\oplus\lambda_b}^i, c)) \oplus L_{c,z_c\oplus\lambda_c}$$
   这里 $L_{c,0} = L_{c,0}^1||\ldots||L_{c,0}^n||0, L_{c,1} = L_{c,1}^1||\ldots||L_{c,1}^n||1$
   将 G_{z_a,z_b} 放在第 $(\lambda_a \oplus z_a, \lambda_b \oplus z_b)$ 行
4. 令输出解密表 $d := \{\lambda_w\}_{w\in Outputs}$
5. 将所有输入线 label $\{L_{w,x_w\oplus\lambda_w}\}_{w\in Inputs}$，输出解密表 d 和所有门的 garbled table $\vec{G}$ 发给 P_1

BMR 协议

设 $F:\{0,1\}^\kappa \to \{0,1\}^{n\kappa}$ 是 PRG，BMR 协议描述如下：

1. 对电路 C 的每条线 $w \in W$，参与方 P_i 随机选 $L^i_{w,0}, L^i_{w,1} \leftarrow \{0,1\}^\kappa$，置换比特 $\lambda^i_w \leftarrow \{0,1\}$。令 $I^i = \{I^i_w\}_{w\in W} = \{F(L^i_{w,0}), F(L^i_{w,1}), L^i_{w,0}, L^i_{w,1}, \lambda^i_w\}_{w\in W}$
2. 每个参与方 P_i 用输入 I^i 调用 $\mathcal{F}^C_{Gen}$，P_1 得到输出 $\{L_{w,x_w\oplus\lambda_w}\}_{w\in Inputs}, \vec{G}, d$
3. P_1 求值 GC：
   1. 对每个电路门 (a,b,c,g)，输入 $L_{a,z_a\oplus\lambda_a} = L^1_a||\ldots||L^n_a||\hat{a}, L_{b,z_b\oplus\lambda_b} = L^1_b||\ldots||L^n_b||\hat{b}$，令 $\hat{a} := lsb(L_{a,z_a\oplus\lambda_a}), \hat{b} := lsb(L_{b,z_b\oplus\lambda_b})$，计算 $L_{c,z_c\oplus\lambda_c} := G_{\hat{a},\hat{b}} \oplus (\bigoplus_{i\in[n]} F(L^i_a, c) \oplus F(L^i_b, c))$
   2. 对输出线 $w \in Outputs, L_w$，令 $x_w := \lambda_w \oplus lsb(L_w)$
4. P_1 将输出 $C(x_1,\ldots,x_n)$ 发给其他参与方

1 介绍

2 BMR 框架
- BMR 协议
- 支持 FreeXOR：[BLO16]

3 参考文献

支持 Free-XOR 的 BMR 协议
Gen 功能定义

设电路 C 线的集合是 W，定义理想功能 $\mathcal{F}_{Gen}^{C}$：

1. 从每个参与方 P_i 接受输入 $I^i = (\Delta^i, \{L^i_{w,0}, \lambda^i_w\}_{w\in W})$
2. 计算 $\Delta := \bigoplus_{i\in[n]} \Delta^i$ 对每个门 (a,b,c,g)，计算
   $\lambda_a := \bigoplus_{i\in[n]} \lambda^i_a$，$\lambda_b := \bigoplus_{i\in[n]} \lambda^i_b$，$\lambda_c := \bigoplus_{i\in[n]} \lambda^i_c$
3. 对每个门 (a,b,c,g)，对 $i \in [n], u, v \in \{0,1\}$ 计算 garbled table：
   $$G^i_{u,v} := (\bigoplus_{j\in[n]} F_{L^j_{a,u}, L^j_{b,v}}(i,c)) \oplus L^i_{c,0} \oplus ((\lambda_a \oplus u)(\lambda_b \oplus v) \oplus \lambda_c)\Delta^i$$
   令 $G_{u,v} := (G^1_{u,v}, \ldots, G^n_{u,v})$
4. 将所有门的 garbled table $\vec{G}$ 发给所有参与方

支持 Free-XOR 的 BMR 协议

离线协议

设 $F:\{0,1\}^{2\kappa}\times[n]\times ID\to\{0,1\}^{\kappa}$ 是双密钥 PRF，离线协议描述如下：

1. 每个参与方 P_i 随机选一个全局差分 $\Delta^i\leftarrow\{0,1\}^{\kappa}$
2. 对电路 C 的每条非 XOR 门输出线 w，参与方 P_i 随机选 $L^i_{w,0}\leftarrow\{0,1\}^{\kappa}$，置换比特 $\lambda^i_w\leftarrow\{0,1\}$。令 $L^i_{w,1}:=L^i_{w,0}\oplus\Delta^i$
3. 对每个 XOR 门 (a,b,c,XOR)，令 $L^i_{c,0}:=L^i_{a,0}\oplus L^i_{b,0},L^i_{c,1}:=L^i_{c,0}\oplus\Delta^i,\lambda^i_c:=\lambda^i_a\oplus\lambda^i_b$
4. 每个参与方 P_i 令 $I^i:=(\Delta^i,\{L^i_{w,0},\lambda^i_w\}_{w\in W})$，调用 $\mathcal{F}^C_{Gen}$，得到输出 $\vec{G}$
5. 对输出线 $w\in Outputs$，P_i 将 λ^i_w 发给所有其他参与方，所有人收到后计算 $\lambda_w:=\oplus_{i\in[n]}\lambda^i_w$
6. 对 P_i 的输入线 w，每个 $P_j,j\neq i$ 将 λ^j_w 发给 P_i，P_i 计算 $\lambda_w:=\oplus_{i\in[n]}\lambda^i_w$ (这一步也可以令所有其他参与方 $P_j,j\neq i$ 选这条线的置换比特时令 $\lambda^j_w=0$)

支持 Free-XOR 的 BMR 协议
在线协议

在线协议描述如下：

1. 发送输入 **label.**
   1. 对 P_i 的输入线 w，设这条线真值是 x_w，P_i 计算 $\hat{w} := x_w \oplus \lambda_w$，并将 $\hat{w}$ 发给所有其他参与方。
   2. 对所有输入线 $w \in Inputs$，所有参与方 P_i 将 $L^i_{w,\hat{w}}$ 发给其他所有参与方
   3. 此时对每个输入线 $w \in Inputs$，每个参与方有 $L^1_{w,\hat{w}}, \ldots, L^n_{w,\hat{w}}$
2. 本地计算 **GC.** 对每个电路门 (a, b, c, g):
   1. 如果 $g = XOR$：每个参与方令 $\hat{c} := \hat{a} \oplus \hat{b}, L^i_{c,\hat{c}} := L^i_{a,\hat{a}} \oplus L^i_{b,\hat{b}}$
   2. 如果 $g = AND$：每个参与方计算 $L^i_{c,\hat{c}} := G^i_{\hat{a},\hat{b}} \oplus (\bigoplus_{j\in[n]} F_{L^j_{a,\hat{a}}, L^j_{b,\hat{b}}}(i, c))$。此时 P_i 得到 $(L^1_{c,\hat{c}}, \ldots, L^n_{c,\hat{c}})$，对比 $L^i_{c,\hat{c}}$ 和自己离线阶段选的 $L^i_{c,0}$ 和 $L^i_{c,1}$，如果等于 $L^i_{c,0}$，则令 $\hat{c} := 0$，否则令 $\hat{c} := 1$
3. 输出重构**.** 对每条输出线 w，各方计算 $x_w := \hat{w} \oplus \lambda_w$

支持 Free-XOR 的 BMR 协议

Gen 协议

为了实现 $\mathcal{F}^C_{Gen}$，就是要计算 garbled table $\vec{G}$，即，对每个 AND 门，计算 $G^i_{u,v} := (\bigoplus_{j\in[n]} F_{L^j_{a,u},L^j_{b,v}}(i,c)) \oplus L^i_{c,0} \oplus ((\lambda_a \oplus u)(\lambda_b \oplus v) \oplus \lambda_c)\Delta^i$

思路：计算加法分享 $G^i_{u,v} = \bigoplus_{j\in[n]} \rho^i_{j,u,v}$，每个参与方 P_j 得到 $\rho^i_{j,u,v}$ 后重构 $G^i_{u,v}$

分解 $G^i_{u,v}$:

- $(\bigoplus_{j\in[n]} F_{L^j_{a,u},L^j_{b,v}}(i,c)) \oplus L^i_{c,0}$：$P_j, j \neq i$ 令自己的分片为 $F_{L^j_{a,u},L^j_{b,v}}(i,c)$，$P_i$ 令自己的分片为 $F_{L^i_{a,u},L^i_{b,v}}(i,c) \oplus L^i_{c,0}$
- $((\lambda_a \oplus u)(\lambda_b \oplus v) \oplus \lambda_c)\Delta^i$:
  1. $(\lambda_a \oplus u)(\lambda_b \oplus v) \oplus \lambda_c$：由于 $(\lambda_a \oplus u)(\lambda_b \oplus v) \oplus \lambda_c = \lambda_a\lambda_b \oplus u\lambda_b \oplus v\lambda_a \oplus uv \oplus \lambda_c$ 各方有 $\lambda_a, \lambda_b, \lambda_c$ 的分享，因此只需计算 $\lambda_a\lambda_b$ 的分享即可，剩下的只需本地做线性组合。由 λ_a, λ_b 的分享计算 $\lambda_a\lambda_b$ 的分享用 GMW 乘法协议 (e.g. OT,HE) 即可完成。
  2. $((\lambda_a \oplus u)(\lambda_b \oplus v) \oplus \lambda_c)\Delta^i$：记为 $r_{c,u,v}\Delta^i$，此时各方已经有了 $r_{c,u,v}$ 的分享，P_i 有 Δ^i，要计算 $r_{c,u,v}\Delta^i$ 的分享，只需 P_i 和其他每个参与方执行一次字符串 OT 即可。

支持 Free-XOR 的 BMR 协议

Gen 协议

具体来说，设分享 $r_{c,u,v} = \bigoplus_{i\in[n]} r^i_{c,u,v}$。$P_i$ 选择一些随机串 $s^j_i \leftarrow \{0,1\}^\kappa, j \neq i$。此时 P_i 与 $P_j, j \neq i$ 执行 OT，P_i 输入 $(s^j_i, s^j_i \oplus \Delta^i)$，$P_j$ 输入 $r^j_{c,u,v}$，P_j 得到输出 $l^i_j := s^j_i \oplus r^j_{c,u,v}\Delta^i$。即 $s^j_i \oplus l^i_j = r^j_{c,u,v}\Delta^i$，因此有

$r_{c,u,v}\Delta^i = \bigoplus_{j\in[n]} r^j_{c,u,v}\Delta^i = \bigoplus_{j\in[n],j\neq i}(s^j_i \oplus l^i_j) \oplus r^i_{c,u,v}\Delta^i$

P_i 令自己的分享为 $\bigoplus_{j\in[n],j\neq i} s^j_i \oplus r^i_{c,u,v}\Delta^i$，$P_j$ 令自己的分享为 l^i_j

支持 Free-XOR 的 BMR 协议

Gen 协议

对于整个 $G_{u,v} = (G^1_{u,v}, \ldots, G^n_{u,v})$ 的分享，只需对 $i \in [n]$，每个 P_i 重复一次上述过程即可得到。

P_1 的分享：

$$(F_{L^1_{a,u},L^1_{b,v}}(1,c) \oplus L^1_{c,0} \oplus r^1_{c,u,v}\Delta^1 \oplus (\bigoplus_{j\in[n],j\neq 1} s^j_1), \quad F_{L^1_{a,u},L^1_{b,v}}(2,c) \oplus l^2_1, \quad \ldots \quad, F_{L^1_{a,u},L^1_{b,v}}(n,c) \oplus l^n_1)$$

P_2 的分享：

$$(F_{L^2_{a,u},L^2_{b,v}}(1,c) \oplus l^1_2, \quad (F_{L^2_{a,u},L^2_{b,v}}(2,c) \oplus L^2_{c,0} \oplus r^2_{c,u,v}\Delta^2 \oplus (\bigoplus_{j\in[n],j\neq 2} s^j_2), \quad \ldots \quad, F_{L^2_{a,u},L^2_{b,v}}(n,c) \oplus l^n_2)$$

$\vdots$

P_n 的分享：

$$(F_{L^n_{a,u},L^n_{b,v}}(1,c) \oplus l^1_n, \quad F_{L^n_{a,u},L^n_{b,v}}(2,c) \oplus l^2_n, \quad \ldots \quad, (F_{L^n_{a,u},L^n_{b,v}}(n,c) \oplus L^n_{c,0} \oplus r^n_{c,u,v}\Delta^n \oplus (\bigoplus_{j\in[n],j\neq n} s^j_n))$$

WRK 协议

[WRK17] 给了一个恶意安全的 BMR 协议构造，这里考虑其协议半诚实安全的形式。在半诚实情况下，他们的构造和 BLO 协议基本一致，唯一的区别是将 n 个短 table 的分享改成了一个长 table 的分享，即，BLO 协议中 P_i 对第 j 个子 table 的分享是用 $F_{L^i_{a,u},L^i_{b,v}}(j,c) \in \{0,1\}^{\kappa}, j \neq i$，而 WRK 协议将每个子 table 合并，$P_i$ 对长 table 的分享是用 $F_{L^i_{a,u},L^i_{b,v}}(c) \in \{0,1\}^{n\kappa}$ 加密的。

WRK 协议

在 WRK 协议中：

P_1 的分享：

$$(F_{L^1_{a,u},L^1_{b,v}}(c) \oplus (L^1_{c,0} \oplus r^1_{c,u,v}\Delta^1 \oplus (\bigoplus_{j\in[n],j\neq 1} s^j_1), \{l^j_1\}_{j\in[n],j\neq 1}))$$

P_2 的分享：

$$(F_{L^2_{a,u},L^2_{b,v}}(c) \oplus (L^2_{c,0} \oplus r^2_{c,u,v}\Delta^2 \oplus (\bigoplus_{j\in[n],j\neq 2} s^j_2), \{l^j_2\}_{j\in[n],j\neq 2}))$$

$\vdots$

P_n 的分享：

$$(F_{L^n_{a,u},L^n_{b,v}}(c) \oplus (L^n_{c,0} \oplus r^n_{c,u,v}\Delta^n \oplus (\bigoplus_{j\in[n],j\neq n} s^j_n), \{l^j_n\}_{j\in[n],j\neq n}))$$

WRK 协议

在 WRK 协议中：

P_1 的分享：

$$(F_{L^1_{a,u},L^1_{b,v}}(c) \oplus (L^1_{c,0} \oplus r^1_{c,u,v}\Delta^1 \oplus (\bigoplus_{j\in[n],j\neq 1} s^j_1), \{l^j_1\}_{j\in[n],j\neq 1}))$$

P_2 的分享：

$$(F_{L^2_{a,u},L^2_{b,v}}(c) \oplus (L^2_{c,0} \oplus r^2_{c,u,v}\Delta^2 \oplus (\bigoplus_{j\in[n],j\neq 2} s^j_2), \{l^j_2\}_{j\in[n],j\neq 2}))$$

$\vdots$

P_n 的分享：

$$(F_{L^n_{a,u},L^n_{b,v}}(c) \oplus (L^n_{c,0} \oplus r^n_{c,u,v}\Delta^n \oplus (\bigoplus_{j\in[n],j\neq n} s^j_n), \{l^j_n\}_{j\in[n],j\neq n}))$$

这里为了和 BLO 协议保持一致，我改变了 WRK 协议的符号，在 WRK 原文中，$F_{L^i_{a,u},L^i_{b,v}}(c) \to H(L^i_{a,u,v}, L^i_{b,u,v}, c), s^j_i \to K_i[r^j_{c,u,v}], l^j_i \to M_j[r^i_{c,u,v}]$

主要参考文献

1. [BLO16] Aner Ben-Efraim and Yehuda Lindell and Eran Omri. Optimizing Semi-Honest Secure Multiparty Computation for the Internet. In CCS 2016.
2. [WRK17] Xiao Wang and Samuel Ranellucci and Jonathan Katz. Global-Scale Secure Multiparty Computation. In CCS 2017.
3. [EKR18] David Evans, Vladimir Kolesnikov, and Mike Rosulek. “A Pragmatic Introduction to Secure Multi-Party Computation”. In: Foundations and Trends in Privacy and Security 2.2-3 (2018), pp. 70–246.
4. [FY22] Dengguo Feng, and Kang Yang. "Concretely efficient secure multi-party computation protocols: survey and more." Security and Safety 1 (2022): 2021001.

参考文献 I

[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*. 1990, pp. 503–513. DOI: 10.1145/100216.100287. URL: https://doi.org/10.1145/100216.100287.

参考文献 II

[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*. 2017, pp. 471–498. DOI: 10.1007/978-3-319-70697-9_17. URL: https://doi.org/10.1007/978-3-319-70697-9_17.

参考文献 III

[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*. 2016, pp. 578–590. DOI: 10.1145/2976749.2978347. URL: https://doi.org/10.1145/2976749.2978347.

参考文献 IV

[Ben+21] Aner Ben-Efraim et al. “Large Scale, Actively Secure Computation from LPN and Free-XOR Garbled Circuits”. 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 III*. Ed. by Anne Canteaut and François-Xavier Standaert. Vol. 12698. Lecture Notes in Computer Science. Springer, 2021, pp. 33–63. DOI: 10.1007/978-3-030-77883-5_2. URL: https://doi.org/10.1007/978-3-030-77883-5_2.

参考文献 V

[EKR18] David Evans, Vladimir Kolesnikov, and Mike Rosulek. "A Pragmatic Introduction to Secure Multi-Party Computation". In: *Foundations and Trends in Privacy and Security* 2.2-3 (2018), pp. 70–246. DOI: 10.1561/3300000019. URL: https://doi.org/10.1561/3300000019.

[FY22] Dengguo Feng and Kang Yang. "Concretely efficient secure multi-party computation protocols: survey and more". In: *Security and Safety* 1 (2022), p. 2021001.

参考文献 VI

[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*. 2017, pp. 598–628. DOI: 10.1007/978-3-319-70694-8_21. URL: https://doi.org/10.1007/978-3-319-70694-8_21.

参考文献 VII

[LSS16] Yehuda Lindell, Nigel P. Smart, and Eduardo Soria-Vazquez. "More Efficient Constant-Round Multi-party Computation from BMR and SHE". In: *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part I*. 2016, pp. 554–581. DOI: 10.1007/978-3-662-53641-4_21. URL: https://doi.org/10.1007/978-3-662-53641-4_21.

参考文献 VIII

[Lin+15] Yehuda Lindell et al. "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*. 2015, pp. 319–338. DOI: 10.1007/978-3-662-48000-7_16. URL: https://doi.org/10.1007/978-3-662-48000-7_16.

[WRK17] 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*. 2017, pp. 39–56. DOI: 10.1145/3133956.3133979. URL: https://doi.org/10.1145/3133956.3133979.

参考文献 IX

[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*. 2020, pp. 1627–1646. DOI: 10.1145/3372297.3417285. URL: https://doi.org/10.1145/3372297.3417285.

Thanks!