通过 Beaver 三元组实现 MPC

《安全多方计算——可证明安全视角》第七章

2026 年 1 月 25 日

目录

1 核心思想与 Beaver 三元组

2 基于 Beaver 三元组的 MPC 协议

3 安全性证明 (UC 框架)

4 Beaver 三元组的生成

MPC 的效率瓶颈与解决方案

背景：计算开销不对称

- **线性门 (加法/常数乘)**：本地计算，开销极低.
- **非线性门 (乘法)**：需要交互，开销巨大.

MPC 的效率瓶颈与解决方案

背景：计算开销不对称

- **线性门 (加法/常数乘)**：本地计算，开销极低.
- **非线性门 (乘法)**：需要交互，开销巨大.

Beaver 的核心思想 (Beaver '91)

离线-在线范式 (Offline-Online Paradigm)：

- **离线阶段**：预计算与输入无关的“相关随机数”(Correlated Randomness).
- **在线阶段**：利用预计算数据，将乘法转化为线性操作，极大提升效率.

Beaver 三元组 (Beaver Triples)

定义: 满足 $c = a \cdot b$ 的随机共享值三元组 $([a], [b], [c])$.

- $a, b \in \mathbb{F}$ 是均匀随机选取的.
- $c = a \cdot b$.
- $[\cdot]$ 表示加法秘密分享.

Beaver 三元组 (Beaver Triples)

定义: 满足 $c = a \cdot b$ 的随机共享值三元组 $([a], [b], [c])$.

- $a, b \in \mathbb{F}$ 是均匀随机选取的.
- $c = a \cdot b$.
- $[\cdot]$ 表示加法秘密分享.

乘法门计算原理: 假设输入为 $[x], [y]$，利用三元组计算 $[xy]$:

1. **掩码计算 (本地)**: $[d] = [x] - [a], \quad [e] = [y] - [b]$.
2. **重构 (交互)**: 公开 d, e (注: d, e 是一次性掩码加密的 x, y，不泄露信息).
3. **结果计算 (本地)**:

$$x \cdot y = (d + a)(e + b) = de + db + ae + ab$$

$$\Longrightarrow [xy] = d \cdot e + d \cdot [b] + e \cdot [a] + [c]$$

协议流程 (Protocol Description)

预设: 参与方持有 m 个 Beaver 三元组 $([a^k], [b^k], [c^k])$.

协议流程 (Protocol Description)

预设: 参与方持有 m 个 Beaver 三元组 $([a^k], [b^k], [c^k])$.

1. 输入分享阶段

P_i 将输入 $x_{i,k}$ 进行加法秘密分享，分发 $[x_{i,k}]$.

协议流程 (Protocol Description)

预设: 参与方持有 m 个 Beaver 三元组 $([a^k], [b^k], [c^k])$.

1. 输入分享阶段

P_i 将输入 $x_{i,k}$ 进行加法秘密分享，分发 $[x_{i,k}]$.

2. 计算阶段 (逐门计算)

- **加法/常数乘**: 本地线性运算 $[x + y] = [x] + [y]$.
- **乘法门 (消耗一个三元组)**:
  - 计算并公开 $d = \mathsf{Open}([x] - [a]), e = \mathsf{Open}([y] - [b])$.
  - 各方本地计算 $[z] = de + d[b] + e[a] + [c]$.

协议流程 (Protocol Description)

预设: 参与方持有 m 个 Beaver 三元组 $([a^k], [b^k], [c^k])$.

1. 输入分享阶段

P_i 将输入 $x_{i,k}$ 进行加法秘密分享，分发 $[x_{i,k}]$.

2. 计算阶段 (逐门计算)

- **加法/常数乘**: 本地线性运算 $[x + y] = [x] + [y]$.
- **乘法门 (消耗一个三元组)**:
  - 计算并公开 $d = \mathsf{Open}([x] - [a]), e = \mathsf{Open}([y] - [b])$.
  - 各方本地计算 $[z] = de + d[b] + e[a] + [c]$.

3. 输出重构阶段

各方发送输出导线的份额 $[y_i]$ 给 P_i 以恢复结果.

安全性分析 (UC Framework)

我们在 $\mathcal{F}_{triple}$-混合模型下证明安全性.

理想功能 $\mathcal{F}_{triple}$

- 生成随机 $a, b \leftarrow \mathbb{F}$，计算 $c = ab$.
- 将 (a_i, b_i, c_i) 发送给 P_i，满足加法分享性质.

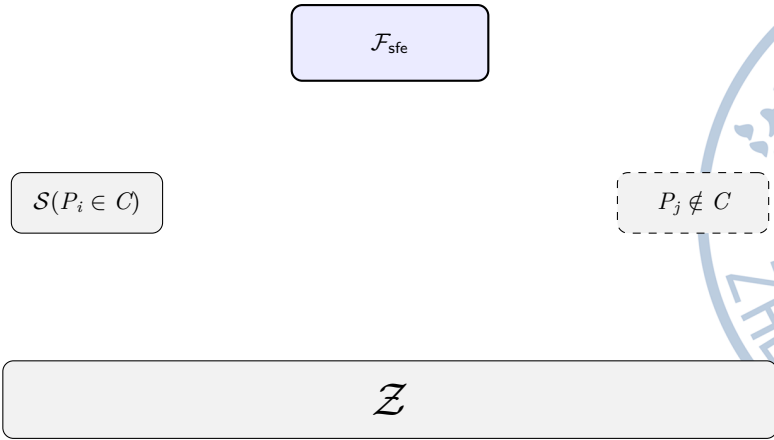
模拟思路 ($\mathcal{S}$):

- **输入阶段**: 模拟器无法获知诚实方输入，用 0 代替模拟.
- **乘法阶段**: 真实协议中 d, e 是随机掩码后的值 (均匀分布). 模拟器可生成随机 d, e 进行模拟，环境无法区分.
- **结论**: 协议实现了对半诚实敌手的 UC 安全.

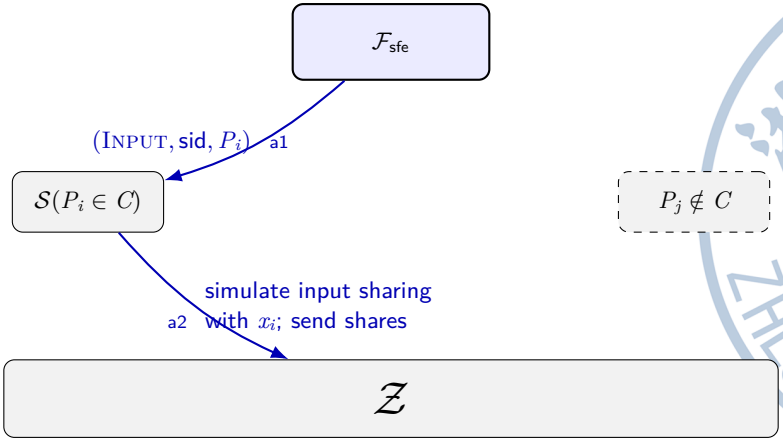
模拟器 $\mathcal{S}$

- 当从 $\mathcal{F}_{\text{sfe}}$ 收到 $(\text{INPUT}, \text{sid}, P_i)$ 时，如果 $P_i \in C$，那么 P_i 是被攻陷的，$\mathcal{S}$ 可以得到 P_i 的输入 $(x_{i,1}, \ldots, x_{i,M_{\text{in}}^i})$. $\mathcal{S}$ 以 $(x_{i,1}, \ldots, x_{i,M_{\text{in}}^i})$ 为输入模拟 P_i 遵循协议执行输入分享.
- 当从 $\mathcal{F}_{\text{sfe}}$ 收到 $(\text{INPUT}, \text{sid}, P_j)$ 时，如果 $P_j \notin C$，那么 P_j 是诚实的. $\mathcal{S}$ 设置 P_j 的输入为 $(x_{j,1}, \ldots, x_{j,M_{\text{in}}^j}) = (0, \ldots, 0)$，然后模拟 P_j 遵循协议执行输入分享，将秘密份额发给被攻陷方.
- 计算阶段，$\mathcal{S}$ 模拟诚实方和被攻陷方遵循协议执行. （也就是说，$\mathcal{S}$ 模拟诚实方以 $(0, \ldots, 0)$ 为输入执行协议）
- 输出重建阶段，重建 P_i 的输出时，$\mathcal{S}$ 发送 $(\text{OUTPUT}, \text{sid}, P_i)$ 给 $\mathcal{F}_{\text{sfe}}$. 如果 P_i 是被攻陷的，$\mathcal{S}$ 将得到 P_i 的输出 $(y_{i,1}, \ldots, y_{i,M_{\text{out}}^i})$. 对于每个输出值 $y_{i,k}$，$\mathcal{S}$ 均匀随机地选择诚实方的份额 $s_{i,k}^{\ell}$，$P_\ell \notin C$，使得 $s_{i,k}^1 + \cdots + s_{i,k}^n = y_{i,k}$. 然后将诚实方的份额发送给 P_i.

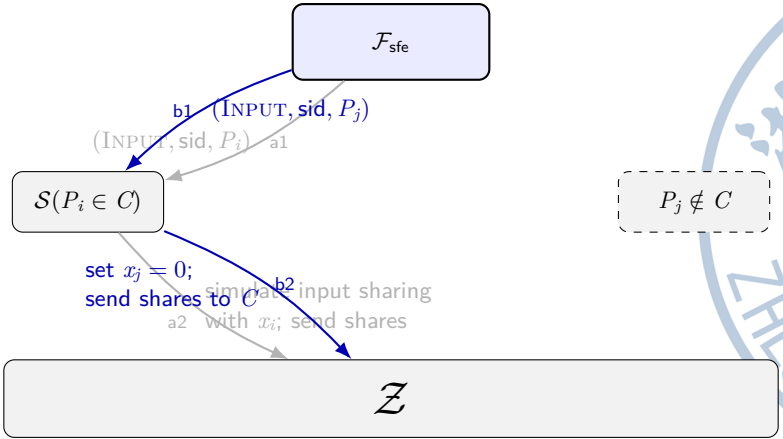
模拟器 $\mathcal{S}$



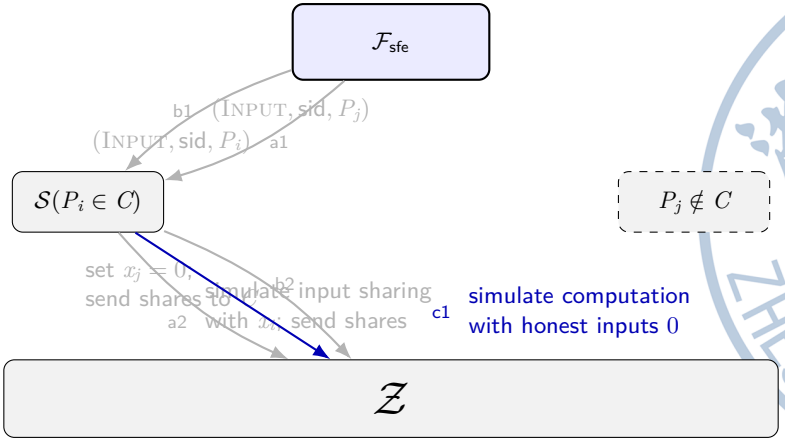
模拟器 $\mathcal{S}$



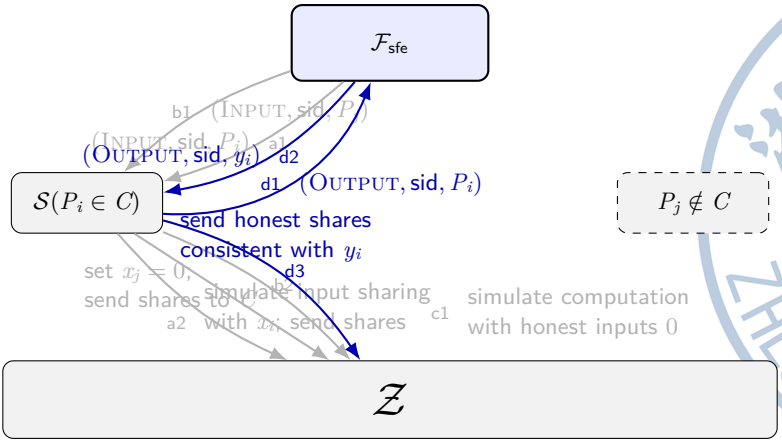
模拟器 $\mathcal{S}$



模拟器 $\mathcal{S}$



模拟器 $\mathcal{S}$



不可区分性

真实世界和理想世界的不同在于：

不可区分性

真实世界和理想世界的不同在于：

1. 理想世界的诚实方输入为 $(0, \ldots, 0)$；

不可区分性

真实世界和理想世界的不同在于：

1. 理想世界的诚实方输入为 $(0, \ldots, 0)$；
2. 输出重建阶段，理想世界的诚实方份额是 $\mathcal{S}$ 模拟的.

不可区分性

真实世界和理想世界的不同在于：

1. 理想世界的诚实方输入为 $(0, \ldots, 0)$；
2. 输出重建阶段，理想世界的诚实方份额是 $\mathcal{S}$ 模拟的.

这对于环境 $\mathcal{Z}$ 是不可区分的，因为：

不可区分性

真实世界和理想世界的不同在于：

1. 理想世界的诚实方输入为 $(0, \ldots, 0)$；
2. 输出重建阶段，理想世界的诚实方份额是 $\mathcal{S}$ 模拟的.

这对于环境 $\mathcal{Z}$ 是不可区分的，因为：

- 加法秘密分享的隐私性确保了秘密份额与诚实方的输入无关；

不可区分性

真实世界和理想世界的不同在于：

1. 理想世界的诚实方输入为 $(0, \ldots, 0)$；
2. 输出重建阶段，理想世界的诚实方份额是 $\mathcal{S}$ 模拟的.

这对于环境 $\mathcal{Z}$ 是不可区分的，因为：

- 加法秘密分享的隐私性确保了秘密份额与诚实方的输入无关；
- 计算乘法门时，因为 Beaver 三元组分享的秘密是均匀随机的，所以秘密重建得到的 $d = w_i - a,\ e = w_j - b$ 是均匀随机值；

不可区分性

真实世界和理想世界的不同在于：

1. 理想世界的诚实方输入为 $(0, \ldots, 0)$；
2. 输出重建阶段，理想世界的诚实方份额是 $\mathcal{S}$ 模拟的.

这对于环境 $\mathcal{Z}$ 是不可区分的，因为：

- 加法秘密分享的隐私性确保了秘密份额与诚实方的输入无关；
- 计算乘法门时，因为 Beaver 三元组分享的秘密是均匀随机的，所以秘密重建得到的 $d = w_i - a,\ e = w_j - b$ 是均匀随机值；
- 输出重建阶段 $\mathcal{S}$ 模拟的份额与真实世界中诚实方的份额也具有相同的分布.

如何生成三元组？(Implementing $\mathcal{F}_{triple}$)

如何生成三元组？(Implementing $\mathcal{F}_{triple}$)

方法一：可信第三方 (TTP)

- 引入半诚实、不合谋的服务器.
- 仅在离线阶段使用，在线阶段无需参与.

如何生成三元组？(Implementing $\mathcal{F}_{triple}$)

方法一：可信第三方 (TTP)

- 引入半诚实、不合谋的服务器.
- 仅在离线阶段使用，在线阶段无需参与.

1. 随机选取 $a, b \leftarrow_{\$} \mathbb{F}$，计算 $c = ab$.
2. 随机选取 $a_1, b_1, c_1, \ldots, a_n, b_n, c_n$，满足 $a = a_1 + \cdots + a_n,\ b = b_1 + \cdots + b_n,\ c = c_1 + \cdots + c_n$.
3. 对所有 $i \in [n]$，将 a_i, b_i, c_i 发送给 P_i.

如何生成三元组？(Implementing $\mathcal{F}_{triple}$)

方法一：可信第三方 (TTP)

- 引入半诚实、不合谋的服务器.
- 仅在离线阶段使用，在线阶段无需参与.

1. 随机选取 $a, b \leftarrow_{\$} \mathbb{F}$，计算 $c = ab$.
2. 随机选取 $a_1, b_1, c_1, \ldots, a_n, b_n, c_n$，满足 $a = a_1 + \cdots + a_n,\ b = b_1 + \cdots + b_n,\ c = c_1 + \cdots + c_n$.
3. 对所有 $i \in [n]$，将 a_i, b_i, c_i 发送给 P_i.

方法二：分布式生成 (MPC)

- 各方本地选择随机 a_i, b_i，$a = \sum a_i, \quad b = \sum b_i$.
- 计算 $[c] = [a \cdot b]$:

如何生成三元组？(Implementing $\mathcal{F}_{triple}$)

方法一：可信第三方 (TTP)

- 引入半诚实、不合谋的服务器.
- 仅在离线阶段使用，在线阶段无需参与.

1. 随机选取 $a, b \leftarrow_{\$} \mathbb{F}$，计算 $c = ab$.
2. 随机选取 $a_1, b_1, c_1, \ldots, a_n, b_n, c_n$，满足 $a = a_1 + \cdots + a_n,\ b = b_1 + \cdots + b_n,\ c = c_1 + \cdots + c_n$.
3. 对所有 $i \in [n]$，将 a_i, b_i, c_i 发送给 P_i.

方法二：分布式生成 (MPC)

- 各方本地选择随机 a_i, b_i，$a = \sum a_i, \quad b = \sum b_i$.
- 计算 $[c] = [a \cdot b]$：
  - 本质是计算乘法.
  - 使用 **GMW (OLE)** 协议在离线阶段完成.

本章总结

- **核心贡献**: 将昂贵的乘法操作移至离线阶段.
- **效率**: 在线阶段仅需简单的线性计算和少量通信，速度极快.
- **通用性**: 适用于任何基于秘密分享的 MPC (GMW, BGW, SPDZ 等).
- **前沿**: 高效生成三元组 (Oblivious Transfer Extension, HE) 是研究热点.

思考题

1. 思考一下 Beaver 三元组是否还有其他生成方法，这些方法的效率如何？同时，它们将引入什么额外的安全假设？
2. 请思考在恶意安全模型中，如何快速高效地批量验证 Beaver 三元组的正确性？
3. 除了 Beaver 三元组，是否还有其他安全多方计算原语可以使用类似的方法在离线阶段先进行预计算，从而大大减少在线阶段的开销？这些原语需要满足什么性质才可以将大部分开销转移到离线阶段？

Q & A