惰性秘密分享及其在 MPC 中的应用

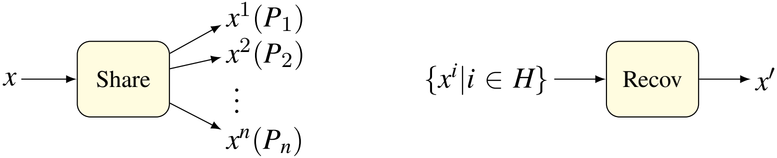
李帅帅

中关村实验室

论文链接：https://eprint.iacr.org/2024/1347

秘密分享（Secret Sharing）

$\mathcal{P} = \{P_1, \ldots, P_n\}$ 为参与方集合；访问结构 $\mathrm{A} \in 2^{\mathcal{P}}$.



- 正确性：如果 $H \in \mathrm{A}$，则 $x' = x$.
- 隐私性：如果 $H \notin \mathrm{A}$，则 $\{x^i | i \in H\}$ 的分布独立于 x.

A 应该是单调的：如果 $H \subseteq H', H \in \mathrm{A}$，则 $H' \in \mathrm{A}$；或者，如果 $H \subseteq H', H' \notin \mathrm{A}$，则 $H \notin \mathrm{A}$.

门限秘密分享（Threshold Secret Sharing）

对于 $t \in [n]$，t-门限秘密分享：A 包括所有大小大于 t 的集合，即

$$\mathrm{A} = \{H \in 2^{\mathcal{P}} : |H| > t\}.$$

门限秘密分享认为所有参与方的权力是平等的：

使用 k 个分片恢复秘密时，能否恢复秘密只和 k 的值有关，而与哪些分片无关.

安全多方计算（Secure Multiparty Computation，MPC）

例

P_1, P_2, P_3 分别有 x, y, z，现在 P_1 想要知道 $f(x, y, z) = x + y + z$.

目标：

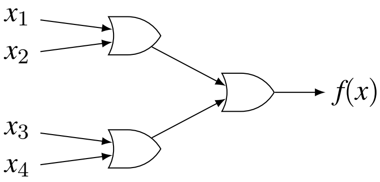
- 正确性：P_1 得到 $x + y + z$.
- 隐私性：敌手无法得到额外的信息.

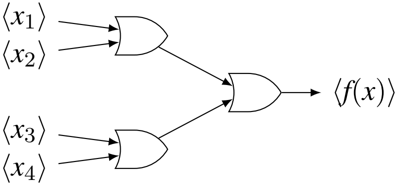
额外的信息：无法从输入和输出推出来的信息.

例如，由于 $(x, x + y + z) \not\to y$，因此 P_1 不应该知道 y 的值.

通用安全计算（Generic/General MPC）：任意多个参与方、任意电路 f.

基于门限秘密分享的 MPC I





计算电路 f 的 MPC 协议如下：

- 输入分享：对每个输入 x，计算 $x \to \langle x \rangle$.
- 电路评估：对每个门 g，计算 $(\langle z_0 \rangle, \langle z_1 \rangle) \to \langle g(z_0, z_1) \rangle$.
- 输出恢复：对输出 $z = f(x)$，计算 $\langle z \rangle \to z$.

基于门限秘密分享的 MPC II

门限秘密分享方案：分享算法 Share 和重构算法 Recov.

- 输入分享：$\langle x\rangle \leftarrow \mathrm{Share}(x)$.
- 输出恢复：$z \leftarrow \mathrm{Recov}(\langle z\rangle)$.
- 核心问题：如何进行电路评估阶段.

考虑到 g 只有两种情况：加法门和乘法门，我们只需要考虑

- 计算 $(\langle z_0\rangle, \langle z_1\rangle) \rightarrow \langle z_0 + z_1\rangle$.
- 计算 $(\langle z_0\rangle, \langle z_1\rangle) \rightarrow \langle z_0 z_1\rangle$.

基于门限秘密分享的 MPC III

使用线性秘密分享

线性秘密分享：如果 f 为一个线性函数，则参与方可以本地计算

$$(\langle x_1\rangle, \ldots, \langle x_k\rangle) \to \langle f(x_1, \ldots, x_k)\rangle.$$

常见方案：加法秘密分享、复制秘密分享、Shamir 分享等.

故设计协议的核心在于

$$\text{计算 } (\langle z_0\rangle, \langle z_1\rangle) \to \langle z_0 z_1\rangle.$$

Goldreich-Micali-Wigderson 协议（STOC 1987）I

设 $\langle x \rangle = (x^1, \ldots, x^n), \langle y \rangle = (y^1, \ldots, y^n)$.

- 基于加法秘密分享方案：$x = x^1 + \cdots + x^n, y = y^1 + \cdots + y^n$.
- 计算加法：$\langle x + y \rangle = (x^1 + y^1, \ldots, x^n + y^n)$.
- 如何计算乘法？注意到

$$xy = \sum_{i \in [n]} x^i y^i + \sum_{(i,j) \in [n]^2}^{i \neq j} x^i y^j.$$

Goldreich-Micali-Wigderson 协议（STOC 1987）II

如果 P_i 和 P_j 生成随机的$s^{i,j}, r^{j,i}$ 使得 $s^{i,j} + r^{j,i} = x^i y^j$，则

$$xy = \sum_{i \in [n]} x^i y^i + \sum_{(i,j) \in [n]^2}^{i \neq j} x^i y^j = \sum_{i \in [n]} (x^i y^i + \sum_{j \in [n] \setminus \{i\}} (s^{i,j} + r^{i,j})) = \sum_{i \in [n]} z_i.$$

使用 OLE（Oblivious Linear Evaluation）协议.

Goldreich-Micali-Wigderson 协议（STOC 1987）III

OLE 协议：两方协议，可以使用同态加密构造。

$$\begin{array}{ccccc} (a,b) & \longrightarrow & \boxed{\text{OLE}} & \longrightarrow & \perp \\ s & \longrightarrow & & \longrightarrow & as+b \end{array}$$

使用 OLE：

$$\begin{array}{ccccc} (x^i,-s^{i,j}) & \longrightarrow & \boxed{\text{OLE}} & \longrightarrow & \perp \\ y^j & \longrightarrow & & \longrightarrow & x^iy^j-s^{i,j}=r^{j,i} \end{array}$$

计算一个乘法门需要 $n(n-1)$ 个 OLE.

Araki-Furukawa-Lindell-Nof-Ohara 协议（CCS 2016）I

三方协议，参与方为 P_1, P_2, P_3.

基于复制（replicated）秘密分享：$x = X_1 + X_2 + X_3$，

$$P_1 \text{ 有 } (X_3, X_2)\text{、} P_2 \text{ 有 } (X_1, X_3)\text{、} P_3 \text{ 有 } (X_2, X_1).$$

设 $\langle x \rangle = (X_3, X_2)(X_1, X_3)(X_2, X_1), \langle y \rangle = (Y_3, Y_2)(Y_1, Y_3)(Y_2, Y_1)$.

- 计算加法：$\langle x + y \rangle = (X_3 + Y_3, X_2 + Y_2)(X_1 + Y_1, X_3 + Y_3)(X_2 + Y_2, X_1 + Y_1)$.
- 如何计算乘法？

Araki-Furukawa-Lindell-Nof-Ohara 协议（CCS 2016）II

注意到

$$xy = \underbrace{(X_3Y_3 + X_3Y_2 + X_2Y_3)}_{Z_3} + \underbrace{(X_1Y_1 + X_1Y_3 + X_3Y_1)}_{Z_1} + \underbrace{(X_2Y_2 + X_2Y_1 + X_1Y_2)}_{Z_2}.$$

保持安全性：使用 0 的分享 $0 = O_1 + O_2 + O_3$，$Z_i \leftarrow Z_i + O_i$.

最后：P_i 将 Z_{i-1} 发送给 P_{i+1}.

基于秘密分享的 MPC 框架优化

当 P_i 生成一个 (t, n) 分享：

任何 t 个分片不泄露秘密的信息.

我们观察到：

如果某 t 个分片包括 P_i 的分片，我们应该允许这 t 个分片泄露秘密.

原因：秘密为 P_i 的输入，我们理应允许 P_i 恢复秘密.

惰性秘密分享（Lazy Secret Sharing）

引入惰性集合 $\mathcal{L} \subseteq [n]$，对 $\mathcal{L}$ 中的分片不做隐私性要求：

- 设 $\langle x \rangle = (x^1, \ldots, x^n)$，对任意 $H \subseteq [n]$，

  放松的隐私性：如果 $|H| \leq t$ 且 $H \cap \mathcal{L} = \varnothing$, 则 $\{x^i\}_{i \in H}$ 的分布独立于 x.
- 注：当 $\mathcal{L} = \varnothing$，惰性分享即为门限秘密分享.

应用：使用惰性分享提升 GMW 和 AFLNO 协议的效率.

惰性加法分享

当 P_1 分享其输入 x:

- 使用加法分享: $\langle x \rangle = (x^1, \ldots, x^n)$, 其中 $x^1, \ldots, x^{n-1}$ 为随机数且 $x^n = x - \sum_{j \in [n-1]} x^j$.
- 使用惰性加法分享: $\langle x \rangle_{\{1\}} = (x, 0, \ldots, 0)$, 惰性集合为 $\mathcal{L} = \{1\}$.

惰性分享可以非交互地生成.

对于 P_i 的输入 x，其分享为 $\langle x\rangle_{\{i\}} = (x^1, \ldots, x^n)$，其中

$$x^j = \begin{cases} x, & \text{if } j = i \\ 0, & \text{if } j \neq i \end{cases}$$

加法同前：每个参与方将自己的分片相加. 例：

- 设 $\langle x\rangle_{\{1\}} = (x, 0, \ldots, 0), \langle y\rangle_{\{2\}} = (0, y, 0, \ldots, 0)$，则

$$\langle x\rangle_{\{1\}} + \langle y\rangle_{\{2\}} = \langle x + y\rangle_{\{1,2\}} = (x, y, 0, \ldots, 0).$$

注意到：

- 惰性集合随着电路深度的增加逐渐增大.
- $\langle x\rangle_{\mathcal{L}_0} + \langle y\rangle_{\mathcal{L}_1} = \langle x + y\rangle_{\mathcal{L}_0 \cup \mathcal{L}_1}$.
- 对于分享 $\langle x\rangle_{\mathcal{L}}$，总是有 $\sum_{j\in\mathcal{L}} x^j = x$ 和 $x^j = 0, j \notin \mathcal{L}$.

计算 $\langle x\rangle_{\mathcal{L}_0}, \langle y\rangle_{\mathcal{L}_1} \to \langle xy\rangle_{\mathcal{L}_0\cup\mathcal{L}_1}$ 需要 P_i 和 P_j 生成 $x^i y^j$ 的一个加法分享：

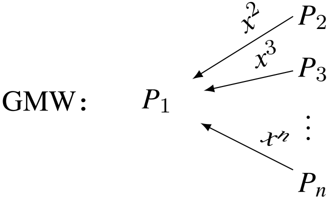
- 对每个 $(i,j) \in [n]^2 \backslash \mathcal{L}_0 \times \mathcal{L}_1$, $x_i y_j = 0 = 0 + 0$.
- 对每个 $(i,j) \in \mathcal{L}_0 \times \mathcal{L}_1$（$i \neq j$），$P_i$ 和 P_j 计算 $x^i y^j = s^{i,j} + r^{j,i}$.

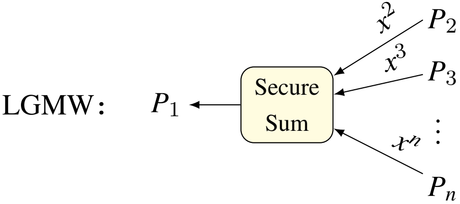
只需要 $|\mathcal{L}_0| \cdot |\mathcal{L}_1| - |\mathcal{L}_0 \cap \mathcal{L}_1|$ 个 OLE.

LGMW：基于惰性加法分享的 GMW IV

输出恢复

假设 P_1 想要得到输出：





- 在 GMW 中，每个分享都是均匀的，因此可以直接收集分片.
- 在 LGMW 中，分享不是均匀的，必须使用安全求和（Secure Sum）.

提高 AFLNO 协议输入分享的效率

当 P_1 分享其输入 x：

- 使用复制秘密分享：$\langle x \rangle = (X_2, X_3)(X_3, X_1)(X_1, X_2)$，其中 X_1, X_2 为随机数且 $X_3 = x - X_1 - X_2$.
- 使用惰性复制分享：$\langle x \rangle = (X_2, X_3)(X_3, 0)(0, X_2)$，其中 X_2 为随机数且 $X_3 = x - X_2$.

通信成本：4 个元素 $\rightarrow$ 2 个元素.

总结

惰性分享的特点:

- 适用性广泛, 不依赖于使用的分享方案.
- 效率提升效果取决于具体的协议.

Q&A

谢谢大家！

ePrint: https://eprint.iacr.org/2024/1347