基于秘密分享方法的安全多方计算协议

冯登国

报告提纲

一、基于秘密分享的MPC协议的基本框架

二、诚实大多数半诚实安全MPC协议

三、诚实大多数假设下乘法验证技术

四、半诚实安全GMW协议及其优化

五、恶意安全SPDZ协议

(n,t)门限线性秘密分享方案的回顾

LSSS定义

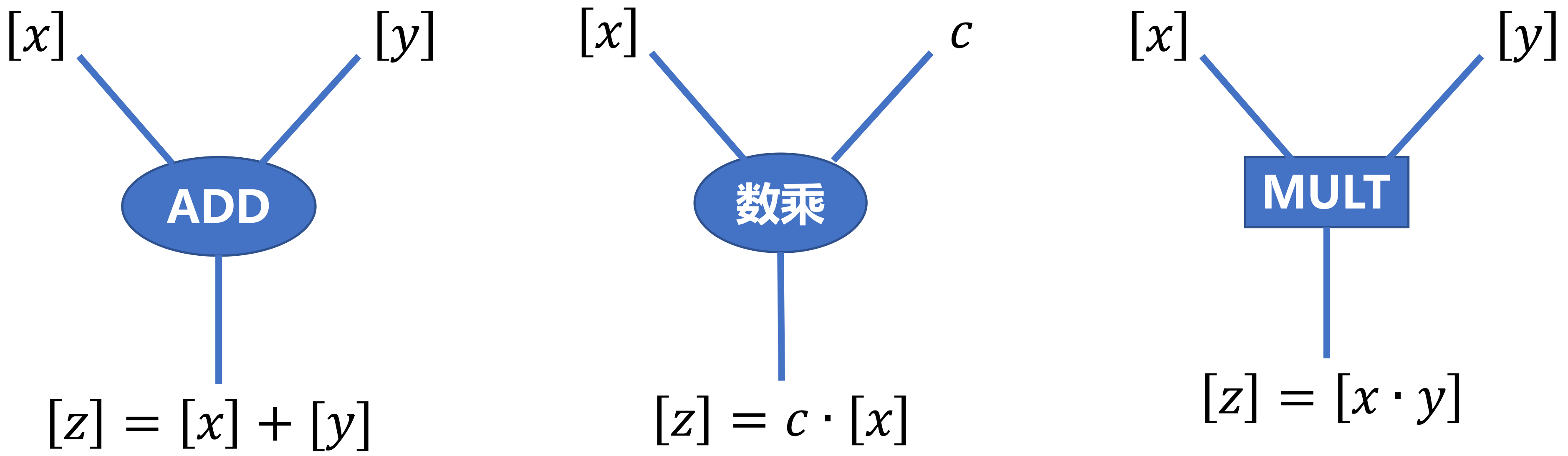
- 分享算法 $[x] \leftarrow Share(x)$：P_i获得份额 x^i, 其中共有n个参与方 $P_1, \cdots, P_n$
- 重构算法 $x \leftarrow Rec([x], i)$：P_i获得秘密值 x （要求至少 $t+1$ 个份额）
- 打开算法 $x \leftarrow Open([x])$：所有参与方获得 x

线性性质

- $[z]=[x]+[y]$，$[z]=[x]+c$，$[z]=c\cdot[x]$ 均能够**本地计算**，**无需通信**，其中 c 为公开的常数
- 应用于MPC协议设计中，加法等线性门是 **free**（即无需通信）

基于线性秘密分享的MPC协议——基本框架（1）

- **分享输入**：对于参与方P_i的每个输入 x，P_i 运行 $[x] \leftarrow Share(x)$ 并将$[x]$ 的份额发送给其他参与方
- **计算电路**：对于每个门的输入分享 $[x],[y]$，计算如下：
  - 对于加法门，**本地计算**输出分享 $[z]=[x]+[y]$(无需通信)
  - 对于数乘门，**本地计算**输出分享 $[z]=c\cdot[x]$(无需通信)
  - 对于乘法门，运行**乘法子协议**计算输出分享 $[z]=[x\cdot y]$
- **重构输出**：对于参与方P_i 的每个电路输出分享 $[z]$，运行分享重构过程 $Rec([z],i)$



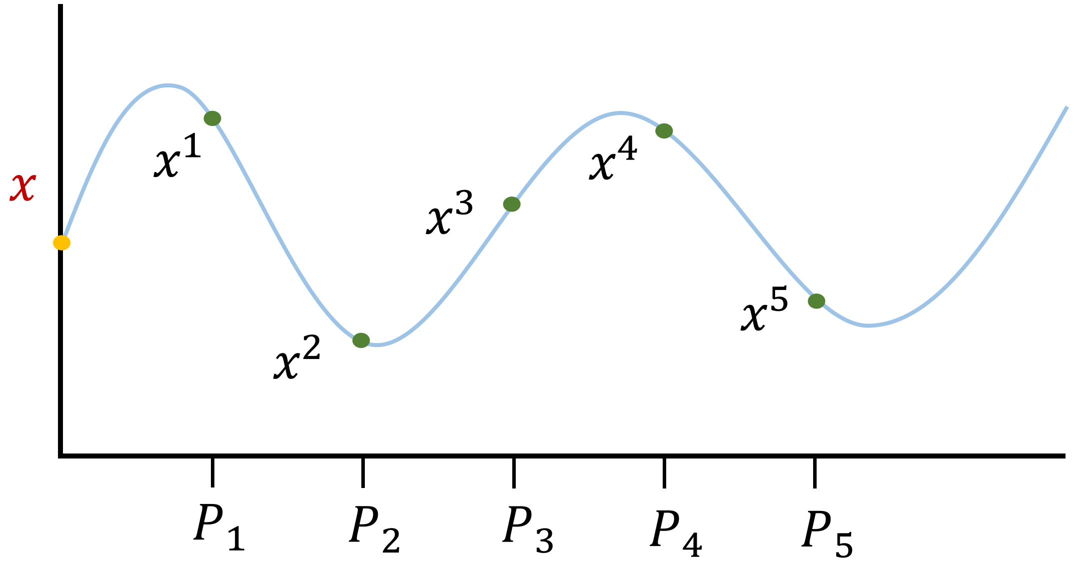
基于线性秘密分享的MPC协议——基本框架（2）

- 支持多个参与方$(P_1, \cdots, P_n)$运行协议；若腐化门限 $t < n/2$，通信复杂度一般为 $O(n)$，若$n/2 \le t < n-1$，通信复杂度一般为 $O(n^2)$
- 加法和数乘等线性门无需通信，计算速度极快
- 协议通信主要发生在乘法门计算
- 所有运算定义在有限域 $\mathbb{F}$上，也可推广至一般环上 (根据应用，环通常考虑剩余类环 $\mathbb{Z}_{2^k}$，由于模运算等价于直接截断)
- 基于线性秘密分享的MPC协议主要考虑如何设计实际高效的乘法子协议

报告提纲

Shamir 秘密分享回顾

- **分享算法** $[x]_t \leftarrow Share(x)$：随机选取次数为$t$的多项式 f 满足 $x = f(0)$，$x^i = f(\alpha_i)$对于$i \in [n]$，其中 $\alpha_1, \cdots, \alpha_n \in \mathbb{F}$为不同的非零域元素
- **重构算法** $x \leftarrow Rec([x]_t)$：需要至少$t+1$个份额，$x = f(0) = \sum \lambda_i(0) \cdot f(\alpha_i)$，其中 $\lambda_i(x) = \Pi_{j \neq i}(x - \alpha_j)/(\alpha_i - \alpha_j)$是拉格朗日系数函数



Shamir 秘密分享生成 —— PRG 优化技术

Shamir 秘密分享另一种生成方式

- $[x]_t \leftarrow Share(x)$：随机选取 t 个域元素 $x^1, \cdots, x^t \in \mathbb{F}$，然后将秘密值 x 和 t个域元素 $x^1, \cdots, x^t$ 通过拉格朗日插值公式获得次数为 t 的多项式 f 满足 $x = f(0)$ 和 $x^i = f(\alpha_i)$
- 其余$t+1$ 个份额计算为 $x^i = f(\alpha_i)$
- P_i 运行该分享算法，将除自己以外的份额发送给所有其他参与方：通信开销为$n-1=2t$个域元素，其中这里不失一般性假设$n=2t+1$

伪随机生成器(PRG)优化

- 指定t个参与方，例如 $P_1, \cdots, P_t$，获得随机份额 $x^1, \cdots, x^t \in \mathbb{F}$
- **Setup阶段**：P_i 发送随机的种子 $\mathrm{seed}_1, \cdots, \mathrm{seed}_t$ 给参与方 $P_1, \cdots, P_t$(可用于计算多个 Shamir 分享，因此通信开销可忽略不计)
- **分享生成阶段**：P_i 和 $P_1, \cdots, P_t$ 计算随机份额 $x^j = PRG(\mathrm{seed}_j)$
- 通信开销降低为 t个域元素 (即通信效率提高了一倍)

随机值的 Shamir 秘密分享生成（1）

随机值的 Shamir 分享生成 —— BGW方法

- 每方P_i随机选取 $r_i \leftarrow \mathbb{F}$，运行 $[r_i]_t \leftarrow Share(r_i)$，将对应份额发送给其他参与方
- 所有参与方本地计算 $[r]_t = [r_1]_t + \cdots + [r_n]_t$

- 诚实参与方选取的随机值 r_i 保证了输出值 r的均匀随机性

随机值的 Shamir 秘密分享生成（2）

范德蒙矩阵批量生成方法

- 每方 P_i随机选取域元素 $u_i \leftarrow \mathbb{F}$，然后运行 $[u_i] \leftarrow Share(u_i)$，并将对应份额发送给其他参与方
- 所有参与方拥有 n 个 Shamir 分享 $[u_1], \ldots, [u_n]$，然后本地计算 $([r_1], \ldots, [r_{n-t}]) = V_{(n-t)\times n} \cdot ([u_1], \ldots, [u_n])$，从而获得 $(n-t)$ 个随机值的 Shamir 秘密分享

范德蒙矩阵定义为$V_{k\times n} = \begin{pmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ \alpha_1^{k-1} & \cdots & \alpha_n^{k-1} \end{pmatrix}$, 其中 $\alpha_1, \ldots \alpha_n \in \mathbb{F}$为互异的非零域元素

- 范德蒙矩阵为超可逆矩阵，即每个$(n-t) \times (n-t)$子矩阵均为可逆矩阵
- 所有输出 Shamir 分享$[r_1], \ldots, [r_{n-t}]$为输入分享 $[u_1], \ldots, [u_n]$ 的线性组合
- $[u_1], \ldots, [u_n]$ 包含$n-t$个诚实参与方计算的 Shamir 分享
- 通过范德蒙矩阵，可将该 $n-t$个 Shamir 分享的随机性扩散至所有输出分享 $[r_1], \ldots, [r_{n-t}]$ 中

半诚实安全的 BGW 乘法子协议

基于 Shamir 秘密分享的 BGW 乘法子协议

- **预处理阶段**：所有参与方生成随机的零分享 $[0]_{2t}$
- **乘法门计算阶段**：给定输入分享$[x]_t, [y]_t$, 计算输出分享 $[\mathrm{z}]_t = [x \cdot y]_t$
  - 所有参与方本地计算 $[\mathrm{z}]_{2t}$，即每方 P_i本地计算份额 $x^i \cdot y^i = h(\alpha_i)$， $h = f \cdot g$为次数 $2t$ 多项式
  - *随机化*：$[\tilde{\mathrm{z}}]_{2t} = [\mathrm{z}]_{2t} + [0]_{2t}$，记对应次数 $2t$ 多项式为 $\tilde{h}$
  - 记 t 次多项式 $\hat{h}$ 由 $\tilde{h}$ 中所有次数 $\leq t$ 单项式组成，存在公共矩阵$\boldsymbol{A}$ (见[BGW88, AL17]) 满足

$$\left(\hat{h}(\alpha_1), \ldots, \hat{h}(\alpha_n)\right)^T = \boldsymbol{A} \cdot \left(\tilde{h}(\alpha_1), \ldots, \tilde{h}(\alpha_n)\right)^T$$

  - *次数约化*：每方 P_i 运行分享算法 $[\tilde{h}(\alpha_i)] \leftarrow Share(\tilde{h}(\alpha_i))$，发送份额给其他参与方，本地计算矩阵乘法，然后运行重构算法$\hat{h}(\alpha_i) \leftarrow Rec([\hat{h}(\alpha_i)])$, 即为 P_i 关于 $[\mathrm{z}]_t$ 的份额 z^i

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). *In STOC 1988*

[AL17] Gilad Asharov and Yehuda Lindell. A Full Proof of the BGW Protocol for Perfectly-Secure Multiparty Computation. In *JoC 2017*

BGW 乘法子协议的 GRR 优化

基于 Shamir 秘密分享的 GRR 乘法子协议

- 给定输入分享$[x]_t, [y]_t$, 计算输出分享 $[z]_t = [x \cdot y]_t$
  - **本地计算**：所有参与方本地计算 $[z]_{2t}$，其中每方 P_i本地计算 $x^i \cdot y^i = h(\alpha_i)$
  - **次数约化步骤 1**：每方 P_i生成 t次秘密分享 $[h(\alpha_i)]_t \leftarrow Share(h(\alpha_i))$，发送相应份额给其他参与方，记对应的$t$次多项式为 q_i
  - **次数约化步骤 2**：每方 P_i本地计算 $z^i = \sum \lambda_j(0) \cdot q_j(\alpha_i)$

$$z = \sum \lambda_j(0) \cdot h(\alpha_j) \quad \Rightarrow \quad [z]_t = \sum \lambda_j(0) \cdot [h(\alpha_j)]_t$$

[GRR98] Rosario Gennaro, Michael O. Rabin and Tal Rabin. Simplified VSS and Fact-Track Multiparty Computations with Applications to Threshold Cryptography. In *PODC 1998*

半诚实安全的 DN 乘法子协议

基于 Shamir 秘密分享的 DN 乘法子协议

➢ **预处理阶段**：生成随机双分享($[r]_t, [r]_{2t}$)，其中 $[r]_d$ 表示次数为 d 的 Shamir 分享

$$([r_1]_t, \cdots, [r_{n-t}]_t)^T = \boldsymbol{V}_{(n-t)\times n} \cdot ([s_1]_t, \cdots, [s_n]_t)^T$$

$$([r_1]_{2t}, \cdots, [r_{n-t}]_{2t})^T = \boldsymbol{V}_{(n-t)\times n} \cdot ([s_1]_{2t}, \cdots, [s_n]_{2t})^T$$

在线阶段

$P_i \neq P_1$		P_1		$P_i \neq P_1$
$[e]_{2t} = [x]_t \cdot [y]_t + [r]_{2t}$	$\xrightarrow{[e]_{2t}}$	$Rec([e]_{2t}) = e = x \cdot y + r$	$\xrightarrow{e}$	$[z]_t = e - [r]_t$

[DN07] Ivan Damgard and Jesper Buus Nielsen. Scalable and unconditionally secure multiparty computation. In *CRYPTO 2007*

优化的 DN 乘法子协议 —— [GSZ20]

基于 Shamir 秘密分享优化的 DN 乘法子协议

➤ **预处理阶段**：生成随机双分享($[r]_t, [r]_{2t}$)，其中 $[r]_d$ 表示次数为 d 的 Shamir 分享

$$([r_1]_t, \cdots, [r_{n-t}]_t)^T = \boldsymbol{V}_{n-t} \cdot ([s_1]_t, \cdots, [s_n]_t)^T$$

$$([r_1]_{2t}, \cdots, [r_{n-t}]_{2t})^T = \boldsymbol{V}_{n-t} \cdot ([s_1]_{2t}, \cdots, [s_n]_{2t})^T$$

在线阶段

$P_i \neq P_1$		P_1		$P_i \neq P_1$
$[e]_{2t} = [x]_t \cdot [y]_t + [r]_{2t}$	$\xrightarrow{[e]_{2t}}$	$Rec([e]_{2t}) = e = x \cdot y + r$	$\xrightarrow{[\boldsymbol{e}]_t}$	$[z]_t = [\boldsymbol{e}]_t - [r]_t$

信息论安全：$t = (n-1)/2$个参与方份额为 0

[GSZ20] Vipul Goyal, Yifan Song, and Chenzhi Zhu. Guaranteed output delivery comes free in honest majority MPC. In *CRYPTO 2020*

优化的 DN 乘法子协议 —— ATLAS（1）

基本思想

- 每个参与方均可作为中心实体（之前协议P_1的角色），**计算 n个乘法门，可以让 n个不同参与方分别作为中心实体**（一个参与方对应一个乘法门）
- 每个中心实体可生成 $e = x \cdot y + r$ 的随机分享（不采用 [GSZ20] 的优化技术）
- **如果中心实体为诚实参与方，那么每个不诚实参与方仅获得一个随机的份额**，没有获得 $e = x \cdot y + r$ 的任何信息
- 计算 n对双分享 $([r]_t, [r]_{2t})$，仅在中心实体为不诚实参与方的情况下需要满足均匀随机性质，即仅需**任意 t对双分享是均匀随机的**

[GLOPS21] Vipul Goyal, Hanjun Li, Rafail Ostrovsky, Antigoni Polychroniadou, and Yifan Song. ATLAS: Efficient and scalable MPC in the honest majority setting. In *CRYPTO 2021*

优化的 DN 乘法子协议 —— ATLAS（2）

基于 Shamir 秘密分享的 ATLAS 乘法子协议

➢ **预处理阶段：生成n对双分享$([r_i]_t, [r_i]_{2t})$，满足其中t 对均匀随机**

$$([s_1]_t, \cdots, [s_t]_t)^T = \boldsymbol{V}_{t\times n} \cdot ([v_1]_t, \cdots, [v_n]_t)^T \qquad ([r_1]_t, \cdots, [r_n]_t)^T = \boldsymbol{V}_{n\times t} \cdot ([s_1]_t, \cdots, [s_t]_t)^T$$

$$([s_1]_{2t}, \cdots, [s_t]_{2t})^T = \boldsymbol{V}_{t\times n} \cdot ([v_1]_{2t}, \cdots, [v_n]_{2t})^T \quad ([r_1]_{2t}, \cdots, [r_n]_{2t})^T = \boldsymbol{V}_{n\times t} \cdot ([s_1]_{2t}, \cdots, [s_t]_{2t})^T$$

在线阶段

$P_i \neq P_{king}$ | P_{king} | $P_i \neq P_{king}$

$$[e]_{2t} = [x]_t \cdot [y]_t + [r]_{2t} \xrightarrow{[e]_{2t}} Rec([e]_{2t}) = e = x \cdot y + r \xrightarrow{[e]_t} [z]_t = [e]_t - [r]_t$$

第 j 次乘法计算由 P_j作为 P_{king}

随机的 Shamir 分享

基于 Shamir 秘密分享半诚实安全乘法子协议的效率比较

乘法子协议	每次乘法每方通信开销 (平均发送域元素数量)		实现电路计算的协议轮数
	信息论安全	计算安全	
[BGW88]	$3(n-1)$	$(n-1)$	$2d+1$
[GRR98]	$n-1$	$(n-1)/2$	d
[DN07]	6	2.5	$2d+1$
[GSZ20]	5.5	2.5	$2d+1$
[GLOPS21]	4	2	$2d+1$
[GLOPS21]	4.5	2.5	$d+1$

DN协议框架

- n : 参与方数量
- d : 电路深度

扩展研读

[GLOPS21] Vipul Goyal, Hanjun Li, Rafail Ostrovsky, Antigoni Polychroniadou, and Yifan Song. ATLAS: Efficient and scalable MPC in the honest majority setting. In *CRYPTO 2021*

报告提纲

诚实大多数假设下乘法子协议性质

$P_i \neq P_{king}$ $\qquad$ P_{king} $\qquad$ $P_i \neq P_{king}$

$$[e]_{2t} = [x]_t \cdot [y]_t + [r]_{2t} \xrightarrow{[e]_{2t}} Rec([e]_{2t}) = e = x \cdot y + r \xrightarrow{[e]_t} [z]_t = [e]_t - [r]_t$$

- 在诚实大多数假设下大部分半诚实安全的乘法子协议（例如上图展示的DN类乘法子协议）满足以下性质
  - **在恶意敌手模型下已经保证了输入/输出的隐私性**
  - 但不能保证计算的正确性，即**允许敌手在输出中引入加法错误**，对于输入分享 $[x], [y]$，输出分享为$[z] = [x \cdot y + d]$，其中d 是敌手选择的加法错误
- 在恶意敌手模型下诚实大多数MPC协议的主要目标是设计高效的乘法验证协议，保证在所有乘法门输出中加法错误为**0**（即没有错误被引入）

乘法正确性验证技术（1）

恶意安全MPC协议 $t < n/2$	乘法验证技术	恶意安全/半诚实安全通信开销比
[LN17]	Beaver 三元组验证技术	7 ×
[CGHIKLN18, NV18]	对偶验证技术	2×
[GSZ20, BGIN20]	分布式零知识证明技术	**1×（最优）**

- [LN17] Yehuda Lindell and Ariel Nof. A Framework for Constructing Fast MPC over Arithmetic Circuits with Malicious Adversaries and an Honest-Majority. In *ACM CCS 2017*
- [CGHIKLN18] 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 *CRYPTO 2018*
- [NV18] Peter Sebastian Nordholt and Meilof Veeningen. Minimising Communication in Honest-Majority MPC by Batchwise Multiplication Verification. In *ACNS 2018*
- [GSZ20] Vipul Goyal, Yifan Song, and Chenzhi Zhu. Guaranteed output delivery comes free in honest majority MPC. In *CRYPTO 2020*
- [BGIN20] Elette Boyle, Niv Gilboa, Yuval Ishai, and Ariel Nof. Efficient fully secure computation via distributed zero- knowledge proofs. In *ASIACRYPT 2020*

乘法正确性验证技术（2）

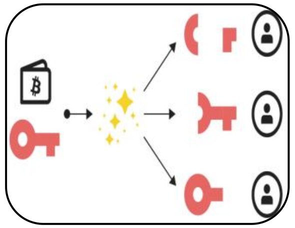
恶意安全MPC协议 $t < n/2$	乘法验证技术	恶意安全/半诚实安全 通信开销比
[LN17]	Beaver 三元组验证技术	7 ×
[CGHIKLN18, NV18]	**对偶验证技术**	2×
[GSZ20, BGIN20]	分布式零知识证明技术	1×（最优）

- [CGHIKLN18] 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 *CRYPTO 2018*
- [NV18] Peter Sebastian Nordholt and Meilof Veeningen. Minimising Communication in Honest-Majority MPC by Batchwise Multiplication Verification. In *ACNS 2018*

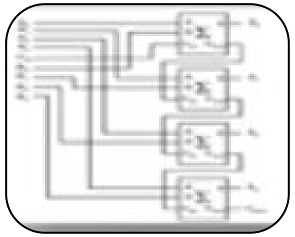
扩展研读

- [BBCGI19] Dan Boneh, Elette Boyle, Henry Corrigan-Gibbs, Niv Gilboa, and Yuval Ishai. Zero-knowledge proofs on secret-shared data via fully linear PCPs. In *CRYPTO 2019*
- [GSZ20] Vipul Goyal, Yifan Song, and Chenzhi Zhu. Guaranteed output delivery comes free in honest majority MPC. In *CRYPTO 2020*
- [BGIN20] Elette Boyle, Niv Gilboa, Yuval Ishai, and Ariel Nof. Efficient fully secure computation via distributed zero- knowledge proofs. In *ASIACRYPT 2020*

对偶验证技术（1）



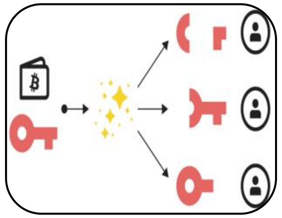
分享输入
(真实输入值x)



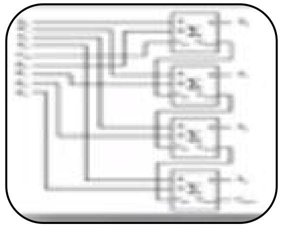
运行半诚实安全协议（1）

验证乘法计算正确性

重构输出



分享输入
(随机化输入 rx)



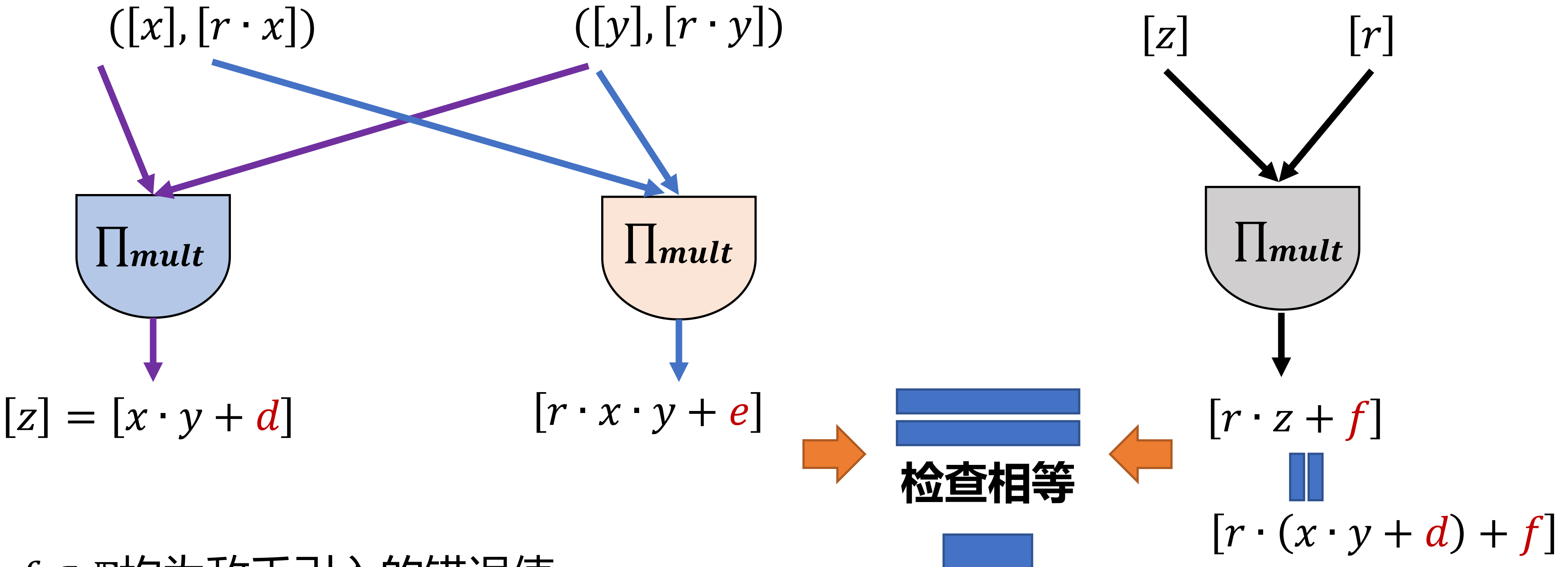
运行半诚实安全协议（2）

[CGHIKLN18] 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 *CRYPTO 2018*

对偶验证技术（2）

随机值的分享生成协议 → **生成一个随机分享 $[r]$**

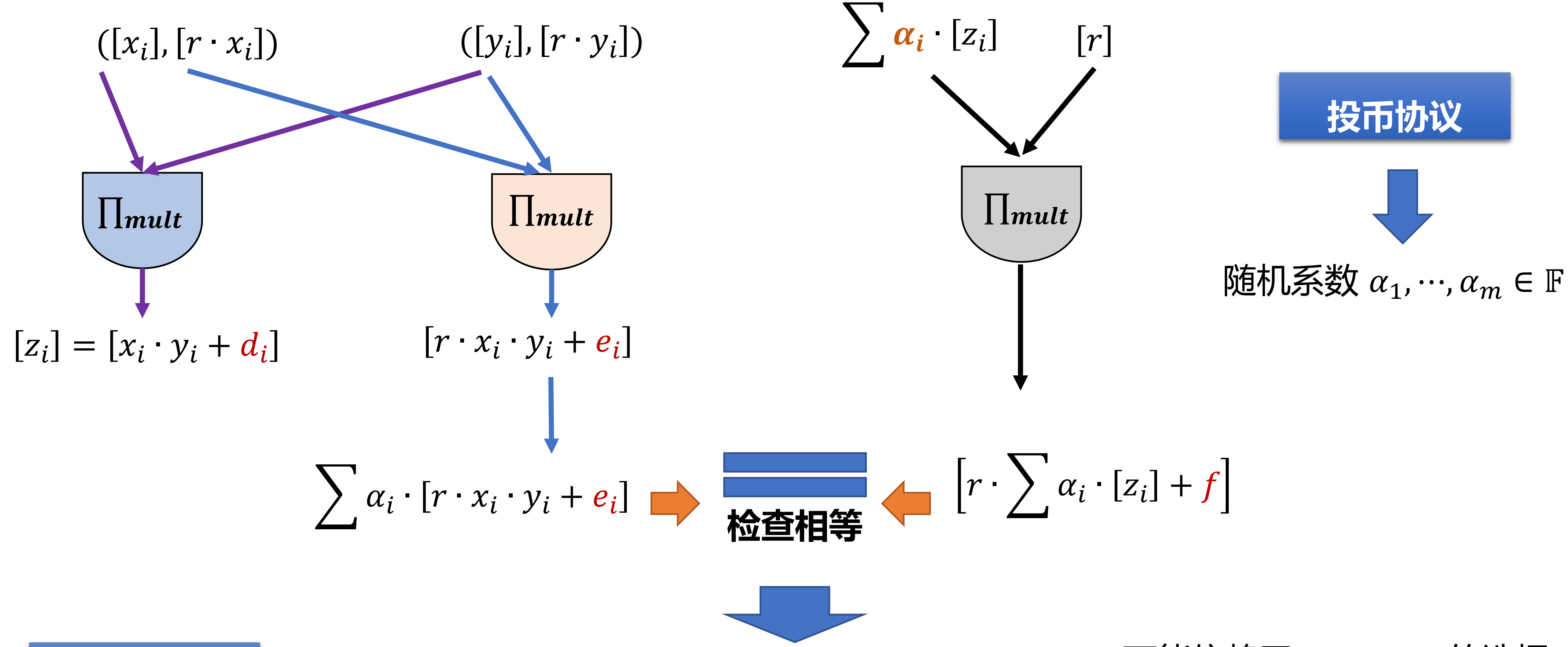
对于每条电路线，所有参与方计算 $([x],[r\cdot x])$，其中 x是真实电路值，利用乘法子协议 $\prod_{mult}$ 计算$[r\cdot x]$



- $d,e,f\in\mathbb{F}$均为敌手引入的错误值
- $r\in\mathbb{F}$均匀随机，敌手未知

$r=(f-e)/d$ → 概率至多$1/|\mathbb{F}|$，域足够大

对偶验证技术（3）

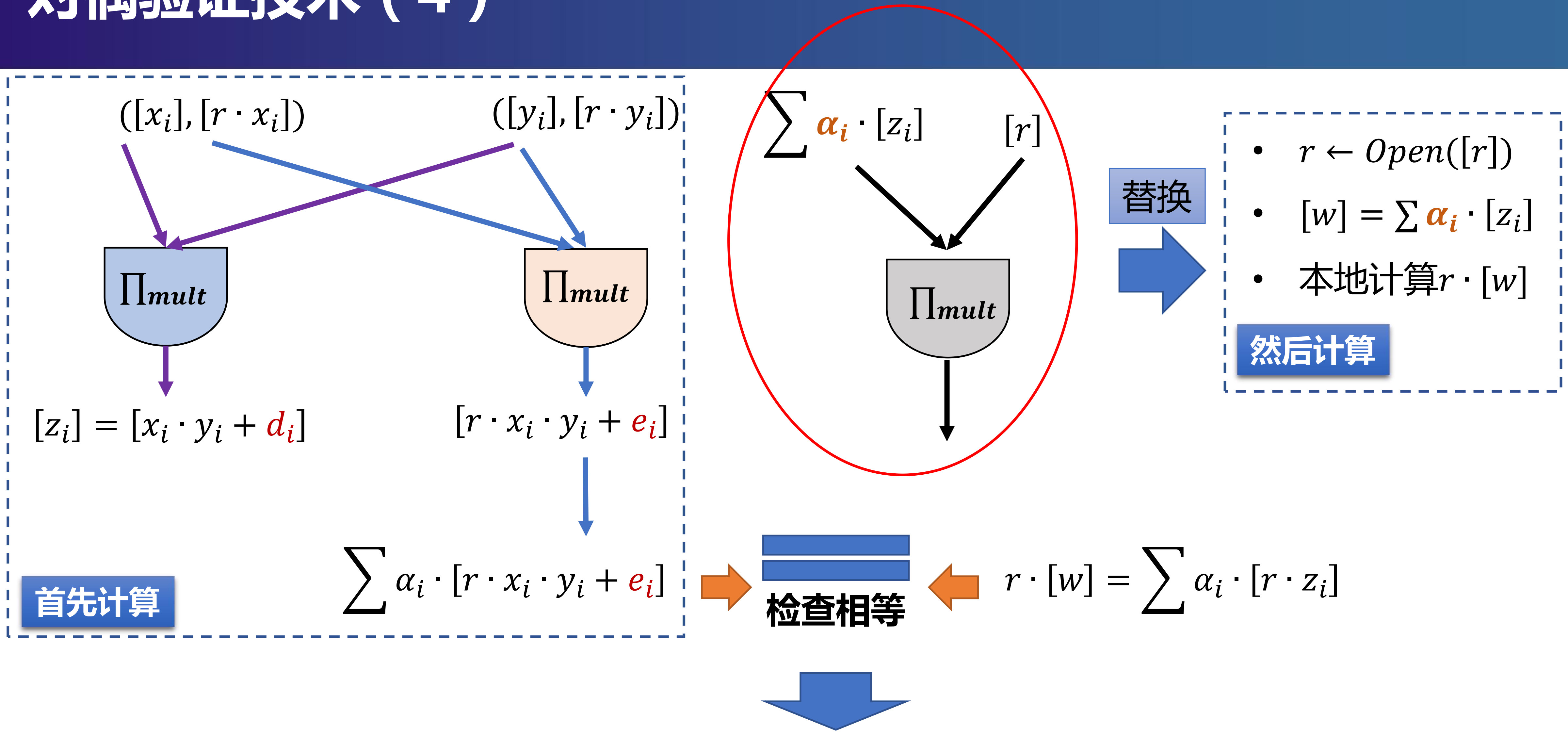


效率优化尝试

$$\sum \alpha_i\cdot(e_i-r\cdot d_i)=f$$

- f 可能依赖于 $\alpha_1,\cdots,\alpha_m$ 的选择
- **无法**推导出 $e_i-r\cdot d_i$=0

对偶验证技术（4）



$$\sum \alpha_i\cdot(e_i-r\cdot d_i)=0 \xrightarrow{\text{概率}1-1/|\mathbb{F}|} (e_i-r\cdot d_i)=0 \xrightarrow{\text{概率}1-1/|\mathbb{F}|} d_i=0$$

对偶验证技术（5）

$[x]$ → 检查相等 ← $[y]$ → 检查 $[z]=[x]-[y]$ 为零分享

随机值的分享生成协议 → 生成一个随机分享 $[r]$

$[z]$ $[r]$

Π_{mult}

$[v]=[r\cdot z+f]$

$v \leftarrow Open([v])$ → 验证$v=0$ → 如果 $z\neq 0$，那么 $r=-f/z$，概率$1/|\mathbb{F}|$

报告提纲

原始的GMW协议

- 针对布尔电路，采用加法秘密分享$[x] = (x_0, x_1)$满足 $x_0 \oplus x_1 = x$
- 利用 1-out-of-4 OT 实现 AND 门计算$z_0 \oplus z_1 = (x_0 \oplus x_1) \wedge (y_0 \oplus y_1)$

a	b
0	0
0	1
1	0
1	1

P_0 P_1

$c_{00}, c_{01}, c_{10}, c_{11}$ $\xrightarrow{}$ OT_1^4 $\xleftarrow{(x_1, y_1)}$

$\mathrm{OT}_1^4 \xrightarrow{z_1 = c_{x_1 y_1}}$

$z_0 \leftarrow \{0,1\}$

$c_{ab} = (x_0 \oplus a) \wedge (y_0 \oplus b) \oplus z_0$

$z_0 \oplus z_1 = (x_0 \oplus x_1) \wedge (y_0 \oplus y_1)$

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *STOC 1987*

优化的GMW协议

- 采用 **1-out-of-2 OT** 替换 1-out-of-4 OT
- 降低单向通信开销至少 **2 倍**

$$z_0 \oplus z_1 = (x_0 \oplus x_1) \wedge (y_0 \oplus y_1)$$

$$= \underbrace{x_0 \wedge y_0 \oplus x_1 \wedge y_1}_{\text{两方本地计算}} \oplus \underbrace{x_0 \wedge y_1 \oplus x_1 \wedge y_0}_{\text{1-out-of-2 OT 计算}}$$

P_0

P_1

$a \leftarrow \{0,1\}$

$a,\ a \oplus x_0 \longrightarrow \mathrm{OT}_1^2 \longleftarrow y_1$

$b = a \oplus (x_0 \wedge y_1) \longrightarrow$

$x_1 \wedge y_0$ 的加法分享能类似计算

GMW协议的推广

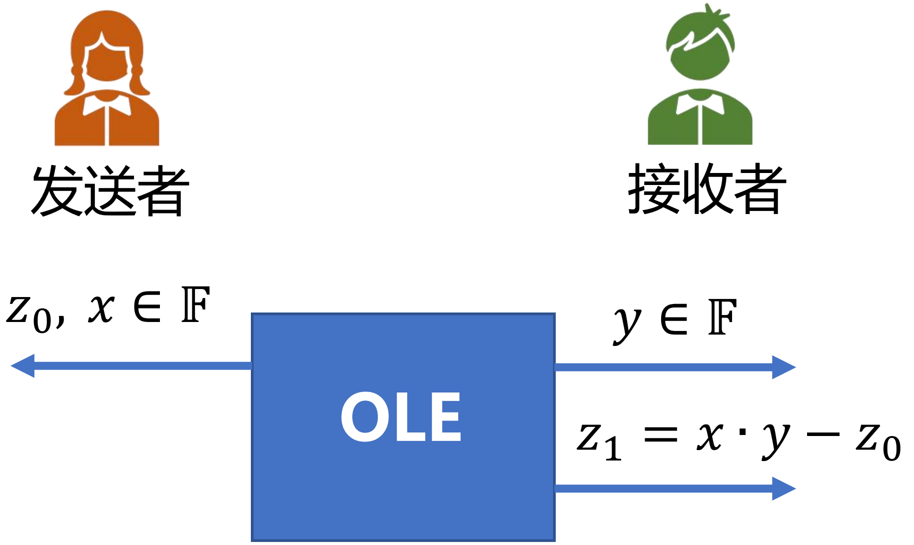
- 以上 GMW 协议仅考虑两个参与方，如何推广至 $n > 2$ 个参与方？
- 如何推广 GMW 协议从布尔电路的分布式计算至算术电路的分布式计算？

$$z_1 \oplus \cdots \oplus z_n = (x_1 \oplus \cdots \oplus x_n) \wedge (y_1 \oplus \cdots \oplus y_n)$$

所有交错项 $x_i \wedge y_j$ 运行 1-out-of-2OT 计算

$$z_1 + \cdots + z_n = (x_1 + \cdots + x_n) \cdot (y_1 + \cdots + y_n) \in \mathbb{F}$$

用 OLE 协议替换 1-out-of-2 OT 协议实现有限域上交错项的安全计算



在线/离线 GMW 协议

Beaver 技术实现在线/离线模式

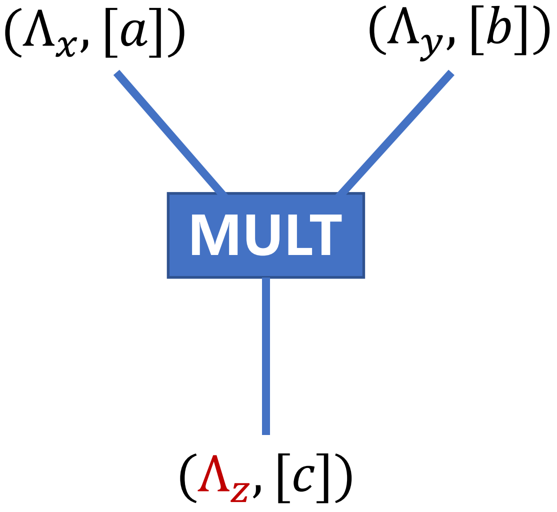
- **电路未知预处理阶段**：运行协议计算随机的 Beaver 三元组 $([a],[b],[c])$，其中$c = a \cdot b$
- **在线阶段**：
  - $\epsilon = Open([x] - [a])$,$\sigma = Open([y] - [b])$
  - $[z] = [c] + \epsilon \cdot [b] + \sigma \cdot [a] + \epsilon \cdot \sigma$

$$z = x \cdot y = (a + \epsilon) \cdot (b + \sigma) = a \cdot b + \epsilon \cdot b + \sigma \cdot a + \epsilon \cdot \sigma = c + \epsilon \cdot b + \sigma \cdot a + \epsilon \cdot \sigma$$

- $\boldsymbol{Open}([\boldsymbol{\epsilon}])$：所有参与方发送 $[\epsilon]$ 的份额给 P_1，P_1 计算 ϵ 并将其发送给其他参与方 $\Rightarrow$ $O(n)$ 通信复杂度，2轮
- $\boldsymbol{Open}([\boldsymbol{\epsilon}])$：所有参与方直接发送 $[\epsilon]$ 的份额给其他参与方 $\Rightarrow$ $O(n^2)$ 通信复杂度，1轮

优化的在线/离线 GMW 协议

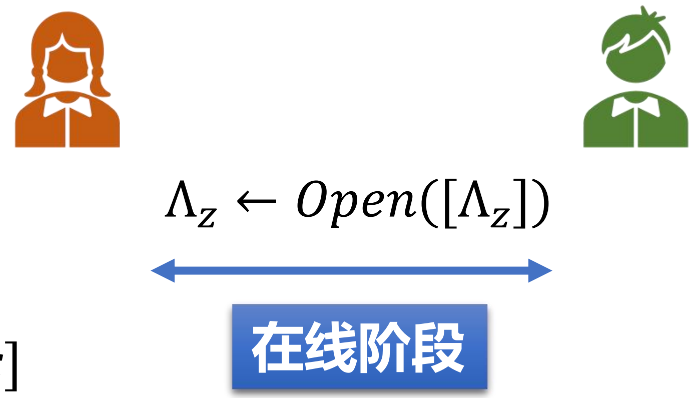
- **电路已知预处理阶段**：计算电路相关的 Beaver 三元组 $([a],[b],[c^*])$，其中 $c^* = a \cdot b$，a 和 b若为乘法门输出，则是随机的域元素，若为加法门输出，则是加法门输入的求和；生成随机加法分享 $[c]$
- 将$(\Lambda_x,[a])$看作秘密x的分享，其中$\Lambda_x = x + a$ 可公开，$[x] = \Lambda_x - [a]$
- 在线通信从 $4(n-1)$ 个元素降低到 $2(n-1)$ 个元素，但离线通信增加 $4(n-1)$ 个元素



$$\Lambda_z = (\Lambda_x - a)\cdot(\Lambda_y - b) + c$$
$$= \Lambda_x \cdot \Lambda_y - \Lambda_x \cdot b - \Lambda_y \cdot a + c^* + c$$

$$[\Lambda_z] = \Lambda_x \cdot \Lambda_y - \Lambda_x \cdot [b] - \Lambda_y \cdot [a] + [c^*] + [c]$$

本地计算，无需通信



[BNO19] Aner Ben-Efraim, Michael Nielsen, and Eran Omri. Turbospeedz: Double your online SPDZ! Improving SPDZ using function dependent preprocessing. In *ACNS 2019*

报告提纲

一、基于秘密分享的MPC协议的基本框架

二、诚实大多数半诚实安全MPC协议

三、诚实大多数假设下乘法验证技术

四、半诚实安全GMW协议及其优化

五、恶意安全SPDZ协议

加强 GMW 协议至恶意安全

- **通用编译器**转化半诚实安全协议为恶意安全协议
  - GMW编译器 [GMW87]，采用零知识证明技术（非黑箱构造）证明每条消息的正确性
  - IPS编译器 [IPS08]，采取两层协议，内层运行半诚实安全不诚实大多数MPC协议，外层运行恶意安全诚实大多数MPC协议（黑箱构造）
    - ✓后续的优化改进 [LOP11,AHIV17,HIMV19,HVW20]
  - 上述编译器的实际效率均偏低

- 目前实际高效不诚实大多数 MPC 协议主要采用 IT-MACs 达到恶意安全性
- 典型 SPDZ 协议，采用 IT-MACs 加强半诚实安全 GMW 协议至恶意安全
- SPDZ 协议容忍腐化门限$t = n - 1$，其中n为参与方数量

信息论消息认证码（IT-MAC）

SPDZ

$$M = x \cdot \Delta$$

- Δ：全局密钥
- x：消息
- M：消息认证码（MAC）

安全性：伪造不同消息$x' \neq x$ 的消息认证码等价于恢复 Δ

$$M = x \cdot \Delta,\ \ M' = x' \cdot \Delta \Rightarrow \Delta = (M - M')/(x - x')$$

SPDZ

$$\sum_{i=1}^{n} M^i = \sum_{i=1}^{n} x^i \cdot \sum_{i=1}^{n} \Delta^i$$

消息认证码、秘密值和全局密钥均被加法分享

表示为 $[\![x]\!] = ([x], [x \cdot \Delta], [\Delta])$

[DPSZ12] Ivan Damg ̊ard, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In *CRYPTO 2012*

SPDZ认证分享的批量打开（1）

加法同态性

$$[\![x]\!] = ([x], [x \cdot \Delta], [\Delta])$$

- $[\![z]\!] = [\![x]\!] + [\![y]\!]$
- $[\![z]\!] = [\![x]\!] + c$
- $[\![z]\!] = c \cdot [\![x]\!]$

均可本地计算

c 为公开的常数

$$[\![x]\!] = ([x], [x \cdot \Delta], [\Delta])$$

$$x \leftarrow Open([\![x]\!])$$

- $x \leftarrow Open([x])$：半诚实安全打开过程
- $CheckZero([\![y]\!] = [\![x]\!] - x)$：检查认证分享值为 0

- $[\![y]\!] = [\![0]\!] \Rightarrow \sum_{i=1}^{n} M^i = 0$（MACs 求和为零）
  - 每方P_i承诺 M^i
  - 每方P_i打开承诺公开M^i，验证$\sum_{i=1}^{n} M^i = 0$

先承诺后打开 ⇒ 抵抗合谋攻击

SPDZ认证分享的批量打开（2）

$Open([\![x_1]\!],\cdots,[\![x_\ell]\!])$：打开 ℓ 个 SPDZ 认证分享

- $(x_1,\cdots,x_\ell) \leftarrow Open([x_1],\cdots,[x_\ell])$：半诚实安全打开过程
- $CheckZero([\![y_1]\!]=[\![x_1]\!]-x_1,\cdots,[\![y_\ell]\!]=[\![x_\ell]\!]-x_\ell)$：**批量验证零认证分享**

- 通过运行投币协议生成随机值 $r_1,\cdots,r_\ell$
  - 投币协议生成随机种子 $seed$，然后利用 $PRG(seed)$ 生成随机值 $r_1,\cdots,r_\ell$
  - 常数通信复杂度
- 随机线性组合：$[\![z]\!]=\sum r_i\cdot[\![y_i]\!]$ (本地计算)
- $CheckZero([\![z]\!])$
- $CheckZero([\![y_1]\!],\cdots,[\![y_\ell]\!])$ **通信复杂度为 $O(1)$，与 ℓ 无关**

SPDZ认证三元组

认证三元组生成是SPDZ协议的效率瓶颈

$(\llbracket x \rrbracket, \llbracket y \rrbracket, \llbracket z \rrbracket)$满足 $z = x \cdot y$

COT VOLE ＋ OT OLE

- 主要分为3种情况
  - 大域 $\mathbb{F}$
  - 二元域$\mathbb{F}_2$
  - 整数环 $\mathbb{Z}_{2^k}$ $(\mathbb{Z} \bmod 2^k)$

- **认证三元组的正确性验证技术**
  - **大域** $\mathbb{F}$ 或 整数环 $\mathbb{Z}_{2^k}$：主要采用**牺牲方法** 等
  - 二元域$\mathbb{F}_2$：主要采用Cut-and-Choose和Bucketing方法、RMFE方法等

认证三元组正确性——牺牲验证技术

验证三元组 $(\llbracket x \rrbracket, \llbracket y \rrbracket, \llbracket z \rrbracket)$，$z = x \cdot y + d$

牺牲三元组 $(\llbracket a \rrbracket, \llbracket y \rrbracket, \llbracket c \rrbracket)$, $c = a \cdot y + e$

- 运行投币协议生成随机值 $r \in \mathbb{F}$
- $\eta \leftarrow Open(r \cdot \llbracket x \rrbracket - \llbracket a \rrbracket)$
- $CheckZero(r \cdot \llbracket z \rrbracket - \llbracket c \rrbracket - \eta \cdot \llbracket y \rrbracket)$

$$
\begin{aligned}
& r \cdot z - c - \eta \cdot y \\
&= r \cdot z - c - (r \cdot x - a) \cdot y \\
&= r \cdot (z - x \cdot y) - (c - a \cdot y) \\
&= r \cdot d - e = 0
\end{aligned}
$$

- $r = e/d \in \mathbb{F}$
- r 均匀随机且 d, e 被确定以后选择
- 概率至多 $1/|\mathbb{F}|$

验证三元组 $(\llbracket x \rrbracket, \llbracket y \rrbracket, \llbracket z \rrbracket)$，$z = x \cdot y + d$

牺牲三元组 $(\llbracket a \rrbracket, \llbracket b \rrbracket, \llbracket c \rrbracket)$, $c = a \cdot b + e$

- 运行投币协议生成随机值 $r \in \mathbb{F}$
- $\eta \leftarrow Open(r \cdot \llbracket x \rrbracket - \llbracket a \rrbracket)$
- $\sigma \leftarrow Open(\llbracket y \rrbracket - \llbracket b \rrbracket)$
- $CheckZero(r \cdot \llbracket z \rrbracket - \llbracket c \rrbracket - \eta \cdot \llbracket y \rrbracket - \sigma \cdot \llbracket a \rrbracket)$

安全性分析方式类似

ℓ 个三元组正确性检查用相同随机数 r，过程类似，$Open$ 和 $CheckZero$采取批量技术

SPDZ协议

- **电路未知预处理阶段：**对于每个乘法门，生成认证三元组 $(\llbracket a\rrbracket, \llbracket b\rrbracket, \llbracket c\rrbracket)$，其中$c = a \cdot b$；对于每条输入线，生成随机认证分享 $\llbracket s\rrbracket$
- **在线阶段——输入处理：**对于 P_i 的输入 x，P_i广播 $\theta = x - s$给其他参与方，所有参与方本地计算$\llbracket x\rrbracket = \llbracket s\rrbracket + \theta$
- **在线阶段——加法门计算：**$\llbracket z\rrbracket = \llbracket x\rrbracket + \llbracket y\rrbracket$
- **在线阶段——乘法门计算：**
  - $\epsilon = Open(\llbracket x\rrbracket - \llbracket a\rrbracket)$,$\sigma = Open(\llbracket y\rrbracket - \llbracket b\rrbracket)$
  - $\llbracket z\rrbracket = \llbracket c\rrbracket + \epsilon \cdot \llbracket b\rrbracket + \sigma \cdot \llbracket a\rrbracket + \epsilon \cdot \sigma$
- **在线阶段——输出重构：**
  - 对于每个电路输出值 y，运行 $y \leftarrow Open(\llbracket y\rrbracket)$
  - 所有 $CheckZero$过程最后一起批量处理（延迟验证）