Trifecta: Faster High-Throughput Three-Party Computation over WAN Using Multi-Fan-In Logic Gates

今天介绍的论文是Sina Faraji和Florian Kerschbaum发表在PoPETS'23上的Trifecta，该工作着眼于三方计算下的布尔电路计算通信开销，提出了新的分享语义支持无需预处理的多输入与门计算，从而减少布尔电路中的AND门深度以减少和交互轮数和通信开销。论文链接如下：
https://petsymposium.org/popets/2023/popets-2023-0107.pdf

本文的相关背景是关于三方计算下的电路深度和通信开销问题，相关的工作专栏之前介绍过，可以参考。

1. 关联随机数

和之前的方案类似，Trifecta也需要生成关联随机数来支撑计算中的分享形式转化。具体来说，本文提出了利用伪随机函数来构造共享随机数，具体的协议如下：

Preprocessing:

(1) **Init**: Each party P_i
- Samples $S_{i,i-1}, S_{i,i+1} \in \{0,1\}^\kappa$
- Sends $S_{i,i-1}$ to P_{i-1} and $S_{i,i+1}$ to P_{i+1}.

(2) **Setup**: Each party P_i
- Sets $\mathbf{R}_{i-1}(x) = \mathbf{F}_{S_{i+1,i}}(x) \oplus \mathbf{F}_{S_{i,i+1}}(x)$
- Sets $\mathbf{R}_{i+1}(x) = \mathbf{F}_{S_{i-1,i}}(x) \oplus \mathbf{F}_{S_{i,i-1}}(x)$

Online:

(3) **GenNextRandom**: Parties P_i and P_j
- Party P_i computes $r_{i,j} = \mathbf{R}_{i-1}(id_{i,j})$
- Party P_j computes $r_{i,j} = \mathbf{R}_{j+1}(id_{i,j})$

(without loss of generality $j = i + 1$)

Figure 1: Correlated randomness functionality

基于上述关联随机数生成的预处理方法，P_i还可以构造两对关联伪随机函数(R_{i-1}, R_{i+1})和(R'_{i-1}, R'_{i+1})，并定义如下掩码函数：

$$
\begin{aligned}
M_{i\to i+1} &= R_{i+1} \\
M_{i\to i-1} &= R_{i-1} \\
M_{i+1\to i-1} &= R_{i-1} \\
M_{i-1\to i+1} &= R'_{i+1}
\end{aligned}
$$

如此，本文定义了如下两方传输协议：

Mask: To send a value v from P_i to P_j

(1) P_i invokes $M_{i\to j}$ to get $m_{i\to j}$.
(2) P_i computes $c = v + m_{i\to j}$ and sends it to P_j.

Figure 2: Message passing with correlated randomness

因为P_k也知道$m_{i\to j}$，所以v在经过上述掩码隐藏并传输之后在(P_j, P_k)之间是两方加法秘密分享。

2. 2-out-of-3秘密分享（π_2^3-sharing）

给定$x \in \mathbb{Z}_{2^n}$，本文提出了一种新的三方秘密分享方案π_2^3-sharing：dealer（在本文中是P_3）随机选取α, β，然后令三方的分享为：

- P_1: $(x+\alpha, \beta)$;
- P_2: $(x+\beta, \alpha)$;
- P_3: (α, β)。

图示表示如下：

	π_2^3-sharing	Rand	Mask
P_1	$(x+\alpha, \beta)$	R_3, R_2	$M_{1\to2}, M_{1\to3}, M_{2\to3}, M_{3\to2}$
P_2	$(x+\beta, \alpha)$	R_1, R_3	$M_{2\to3}, M_{2\to1}, M_{3\to1}, M_{1\to3}$
P_3	(α, β)	R_3, R_1	$M_{3\to1}, M_{3\to2}, M_{1\to2}, M_{2\to1}$

Table 1: Different shares and PRFs held by the parties

针对布尔电路（即$n=1$），本文接下来构造了各种优化电路。为了方便起见，后续的符号“+”表示XOR，·或者省略乘法表示AND。

3. 基础协议构造

首先介绍基础的XOR和2-输入AND计算。给定x_1的分享$\{(x_1+\alpha_1, \beta_1), (x_1+\beta_1, \alpha_1), (\alpha_1, \beta_1)\}$和$x_2$的秘密分享$\{(x_2+\alpha_2, \beta_2), (x_2+\beta_2, \alpha_2), (\alpha_2, \beta_2)\}$，相关协议构造如下。

3.1 XOR 计算

XOR门的计算不需要交互，三方可以计算各自的计算流程如下：

- P_1: 计算并输出$\{(x_1+x_2)+(\alpha_1+\alpha_2), (\beta_1+\beta_2)\}$;
- P_2: 计算并输出$\{(x_1+x_2)+(\beta_1+\beta_2), (\alpha_1+\alpha_2)\}$;
- P_3: 计算并输出$\{(\alpha_1+\alpha_2), (\beta_1+\beta_2)\}$

很容易验证上述输出可以构成x_1+x_2的π_2^3-sharing。

3.2 2-输入 AND 计算

2输入-AND门的计算则稍微复杂一些，需要1轮交互。具体计算需要分三步展开：

1. **Step 1:** 三方计算3-out-of-3 sharing如下：
   - P_1计算$v_1 = (x_1+\alpha_1)(x_2+\alpha_2)$;

- P_2计算$v_2 = (x_1 + \beta_1)\alpha_2 + (x_2 + \beta_2)\alpha_1$;
- P_3计算$v_3 = \alpha_1\beta_1 + \alpha_2\beta_2 + \alpha_1\alpha_2$

2. 有了上述3-out-of-3 sharing，三方通信如下：
   - P_1发送$c_1 = v_1 + m_{1\to2}$给P_2;
   - P_2发送$c_2 = v_2 + m_{2\to1}$给P_1;
   - P_3发送$c_3 = v_3 + m_{3\to1}$给P_1.
3. 最后，三方构造x_1x_2的π_2^3-sharing如下：
   - P_1计算$(t + \alpha_t, \beta_t) = (v_1 + c_2 + c_3, c_3 + m_{1\to2})$;
   - P_2计算$(t + \beta_t, \alpha_t) = (v_2 + c_1 + m_{3\to1}, m_{2\to1} + m_{3\to1})$;
   - P_3计算$(\alpha_t, \beta_t) = (m_{2\to1} + m_{3\to1}, c_3 + m_{1\to2})$.

可以验证，$(t + \alpha_t) + \alpha_t = (t + \beta_t) + \beta_t = x_1x_2$，因此最终结果是$\pi_2^3$-sharing的形式。

4. 多输入 AND 计算

对于多输入AND门的计算，之前的ABY2.0和Meteor分别在两方、三方下进行了探索，但是这些工作都需要大量的预计算开销。本文提出的方案不需要预计算（除了关联伪随机函数构造）。

4.1 3-输入 AND

给定x_1, x_2, x_3的秘密分享，在本文的秘密分享语义下求$t = x_1x_2x_3$的关键在于在P_1和(P_2, P_3)之间求的两方加法秘密分享$(t + \alpha_t, \alpha_t)$；对称的，也需要在P_2和(P_1, P_3)之间求的两方加法秘密分享$(t + \beta_t, \beta_t)$。为了构造出上述的秘密分享形式，本文首先提出了构造两方加法秘密分享 $(t + \alpha_t, \alpha_t)$ 的协议如下：

- **Compute (2,2)-sharing of $x_1x_2x_3$**

(1) P_2 computes

$$\mathbf{v}_2 = \begin{cases} v_2^1 = (x_1+\beta_1)(x_2+\beta_2)(x_3+\beta_3) \\ v_2^2 = (x_1+\beta_1)(x_2+\beta_2) \\ v_2^3 = (x_1+\beta_1)(x_3+\beta_3) \\ v_2^4 = (x_2+\beta_2)(x_3+\beta_3) \end{cases}$$

(2) P_2 sends $c_2^i = v_2^i + m_{2\to1}^i$ to P_1 for all v_2^i.

(3) P_1 computes

$$\begin{aligned} t+\alpha_t = {} & c_2^1 \\ & + c_2^2\beta_3 + c_2^3\beta_2 + c_2^4\beta_1 \\ & + (x_1+\alpha_1)\beta_2\beta_3 + \beta_1(x_2+\alpha_2)\beta_3 \\ & + \beta_1\beta_2(x_3+\alpha_3) \\ & + m_{3\to2}^1 \end{aligned}$$

(4) P_3 computes

$$\begin{aligned} v_3^1 = {} & m_{2\to1}^1 + m_{2\to1}^2\beta_3 + m_{2\to1}^3\beta_2 + m_{2\to1}^4\beta_1 \\ & + \alpha_1\beta_2\beta_3 + \beta_1\alpha_2\beta_3 + \beta_1\beta_2\alpha_3 \end{aligned}$$

(5) P_3 sends $c_3^1 = v_3^1 + m_{3\to2}^1$ to P_2 and sets $\alpha_t = c_3^1$.

(6) P_2 sets $\alpha_t = c_3^1$.

Figure 3: Computing (2,2)-sharing of the product $x_1x_2x_3$

上述协议只需要在第(2)和(5)两步进行通信，且两次通信可以并行。同时，对于$(t+\beta_t, \beta_t)$也可以并行执行，因此计算3输入与门本文只需要1轮通信。

4.2 ℓ-输入 AND

上述3输入的计算方式可以自然推广到对于多输入AND门的计算。本文也是具体描述了求$(t+\alpha_t, \alpha_t)$的子协议如下：

- **Compute (2,2)-sharing $x_1 \ldots x_\ell$**

(1) P_2 computes

$$v_{\phi(I)} = \prod_{x_i \in I} (x_i + \beta_i)$$

for $I \subseteq \mathcal{P}(X)$ and $|I| > 1$,

(2) P_2 sends $c_2^{\phi(I)} = v_2^{\phi(I)} + m_{2\to 1}^{\phi(I)}$ to P_1 for all $v_2^{\phi(I)}$.

(3) P_1 computes

$$t + \alpha_t = \sum_{I \subseteq X, |I|>1} c_2^{\phi(I)} \prod_{x_j \notin I} \beta_j + \sum_{x_i \in X, I=\{x_i\}} (x_i + \alpha_i) \prod_{x_j \notin I} \beta_j + b \prod_{i \in \mathbb{L}} \beta_i + m_{3\to 2}^1$$

where $b = 1$ if n is even o.w. $b = 0$.

(4) P_3 computes

$$v_3^1 = \sum_{I \subseteq X, |I|>1} m_{2\to 1}^{\phi(I)} \prod_{x_j \notin I} \beta_j + \sum_{x_i \in X, I=\{x_i\}} \alpha_i \prod_{x_j \notin I} \beta_j$$

(5) P_3 sends $c_3^1 = v_3^1 + m_{3\to 2}^1$ to P_2 and sets $\alpha_t = c_3^1$.

(6) P_2 sets $\alpha_t = c_3^1$.

Figure 4: Computing (2,2)-sharing of the product $x_1 \ldots x_\ell$

其中，P_2的计算量和通信量均是输入个数的组合数（$2^\ell - \ell - 1$）。

上述方案的正确性和安全性均很容易验证。

5. 常见电路协议优化

基于上述对于多输入AND门的优化，本文提出了针对布尔电路加法器、乘法器和比较电路的优化。为了平衡通信轮数和通信量，本文令$\ell \leq 8$。可以从下表的理论分析结果看出，本文提出的方案大大提升了现有布尔电路的运行效率。

Algorithm	depth				size			
	16	32	64	128	16	32	64	128
Ripple-Carry	16	32	64	128	31	63	127	255
Sklansky	5	6	7	8	65	161	385	897
ℓ ≤4-fan-in	3	4	5	8	73	177	433	993
ℓ ≤8-fan-in	3	3	3	4	87	213	561	1249

Table 2: Comparison of adder circuit depth and size for different constructions and bit-widths

Algorithm	depth				size			
	16	32	64	128	16	32	64	128
Standard	45	93	189	381	496	2016	8128	32640
Wallace	13	15	18	21	512	2058	8226	32836
ℓ ≤4-fan-in	7	9	9	11	1229	5304	21997	89416
ℓ ≤8-fan-in	6	6	8	11	1422	6612	24834	96679

Table 3: Comparison of multiplier circuit depth and size for different constructions and bit-widths

Algorithm	depth				size			
	16	32	64	128	16	32	64	128
Standard	5	6	7	8	63	143	319	703
ℓ ≤4-fan-in	3	4	4	5	39	95	207	479
ℓ ≤8-fan-in	3	3	3	4	37	83	175	415

Table 4: Comparison of comparator circuit depth and size for different constructions and bit-widths

6. 实验评估

本文对布尔电路下的加法、乘法和比较，已经AES安全计算等任务，基于本文的秘密分享进行实现，并和之前的方案进行了对比。结果显示，本文在多种任务上都优于之前的方案。

Func.	Work	Batch = 1				Batch = 100			
		Sim.		WAN		Sim.		WAN	
		32	64	32	64	32	64	32	64
Add.	[4]	0.150	0.150	0.150	0.207	0.583	0.652	1.211	1.37
	This	0.212	0.213	0.152	0.204	0.252	0.252	0.252	0.253
Mult.	[4]	0.586	0.639	1.179	1.316	7.493	11.375	20.35	24.64
	This	0.767	0.925	0.461	0.628	0.960	1.422	0.727	1.174
Comp.	[4]	0.150	0.199	0.200	0.351	0.651	0.874	1.448	3.279
	This	0.211	0.212	0.154	0.155	0.251	0.252	0.251	0.249

Table 8: Comparison of online running time (sec) of our protocol and Beaver et al.'s protocol [4] for addition, multiplication and comparison circuits

Param.	Work	AES-128		AES-192		AES-256	
		Sim.	WAN	Sim.	WAN	Sim.	WAN
Runtime (sec)	[42]	2.006	2.052	2.408	2.455	2.810	2.856
	[1]	1.964	2.071	2.417	2.520	2.720	2.885
	[4]	0.651	1.147	0.673	1.120	0.677	1.124
	This	1.056	0.655	1.258	0.747	1.458	0.858
Comm (KB)	[42]	2.64		3.16		3.69	
	[1]	2.04		2.42		2.85	
	[4]	11680		13080		16110	
	This	165.7		198.2		230.8	

Table 11: Comparison of online running time and communication cost of our protocol and [1, 4] for AES. Results of our protocol and [1, 42] are reported for 100 parallel instances

本篇博客只介绍了协议的设计思路和部分实验结果。协议的具体正确性推导和更多实验评估请参考原文。