本次介绍的是微软Deevashwer Rathee等人发表在CCS'2020的工作 CrypTFlow2: Practical 2-Party Secure Inference。

0. Background & Motivation

安全两方计算（2PC）下神经网络推理预测一直是一个研究难题，其难点一是非线性计算的开销巨大，例如比较、除法；另一个则是计算精度的损失。CrypTFlow2基于茫然传输（Oblivious Transfer, OT）提出了安全比较的一种新的协议，并对该协议进行了深度优化。进一步，利用该比较协议设计了面向神经网络的多个算子协议，例如ReLU、Truncation、faithful Division (divisor is public), Avgpool、和Maxpool等。并且，本文的针对环 $\mathbb{Z}_L(L=2^\ell)$ 和 $\mathbf{Z}_n$ (n为任意大整数)设计了两个版本，从而适配面向OT和同态加密（Homomorphic Encryption, HE）的线性层计算。最后，本文代码已经开源，链接 https://github.com/mpc-msri/EzPC/tree/master/SCI。

1. Premliminaries

本文面向两方计算下的半诚实敌手，主要用到两方下的加性秘密分享，OT，基于OT构造的AND 三元组、Multiplexer、和B2A转化，以及HE，其中秘密分享已经很熟了，而HE主要用在神经网络线性层。本文的核心构造是基于OT的，我们主要在这里在回顾一下OT方面的知识。

1.1 Oblivious Transfer

$\binom{k}{1}-\mathrm{OT}_\ell$ 表示 k 选1的OT：发送方（sender）有 k 条信息 $m_0,...m_{k-1}$, 每条数据长度为 ℓ，执行完毕后接收到根据自己的选择 i 只获得 m_i。$k=2$ 则是2选1 OT。进一步，关联OT（correlated OT）$\binom{2}{1}-\mathrm{COT}_\ell$ 则是发送方输入 x ，输入方根据自己的选择比特 b 得到 r 或者 $x+r$ ，而发送方获得 r。$\binom{k}{1}-\mathrm{OT}_\ell$ 和 $\binom{2}{1}-\mathrm{COT}_\ell$ 都需要两轮通信，通信量分别为 $2\lambda+k\ell$ 和 $\lambda+\ell$。而一般的 $\binom{2}{1}-\mathrm{OT}_\ell$ 通信量为 $\lambda+2\ell$。其中 λ 为 $\binom{2}{1}-\mathrm{OT}_\ell$ 的安全参数，一般 $\lambda=128$。更多关于OT的理论知识，可以参考[]

1.2 OT-based AND triples，Muxltiplexer，& B2A

AND Triples：$\mathcal{F}_{\mathrm{AND}}$ 可以用 AND 三元组计算。生成三元组 $(\langle d\rangle_b^B,\langle e\rangle_b^B,\langle f\rangle_b^B)$ 满足 $f=de$ 一般用OT。传统的方法一次 OT 生成一个三元组，本文使用 $\binom{16}{1}-\mathrm{OT}_2$，一次OT产生两个三元组。粗略来说两个对每个三元组随机算则 $\langle d\rangle_b^B,\langle e\rangle_b^B \xleftarrow{\$} \{0,1\}$ 。P_1 作为接收者设置前两个比特位为 $\langle d\rangle_1^B||\langle e\rangle_1^B$，针对第二个三元组的后两个比特位设置类似；发送者 P_0 对于第 i 条信息第一个比特 $i\in\{0,1\}^4$，设置为 $r\oplus((i_1\oplus\langle d\rangle_0^B)\wedge(i_2\oplus\langle e\rangle_0^B))$ 其中 $i=i_1||i_2$。对于第二个元组，设置第 i 条信息的第2个比特位类似。从而，P_0 最终获得 $\langle f\rangle_0^B=r$，而 P_1 获得 $\langle f\rangle_1^B=\langle f\rangle_0^B\oplus(d\wedge e)$ 。如此，一共通信 $2\lambda+16$，均摊下来对于每一个元组通信 $\lambda+16$ 比特。

Multiplexer: $\mathcal{F}_{\text{MUX}}$ 输入 $\langle a\rangle^n$ 和 $\langle c\rangle^B$，如果 $c=1$ 则输出 $\langle a\rangle^n$；否则输出 0。如算法6所示，根据 $\binom{2}{1}-\text{OT}_\eta$，$x_1=-r_0+c\cdot\langle a\rangle_0^n$，$x_0=-r1+c\cdot\langle a\rangle_1^n$。如此 $z=z_0+z_1=c\cdot a$。由于调用了两次OT，通信为 $2(\lambda+2\eta)$。

```
Algorithm 6 Multiplexer, $\Pi^n_{\text{MUX}}$:

Input: For $b \in \{0,1\}$, $P_b$ holds $\langle a\rangle^n_b$ and $\langle c\rangle^B_b$.
Output: For $b \in \{0,1\}$, $P_b$ learns $\langle z\rangle^n_b$ s.t. $z = a$ if $c = 1$, else $z = 0$.

1: For $b \in \{0,1\}$, $P_b$ picks $r_b \xleftarrow{\$} \mathbb{Z}_n$.
2: $P_0$ sets $s_0, s_1$ as follows: If $\langle c\rangle^B_0 = 0$, $(s_0, s_1) = (-r_0, -r_0 + \langle a\rangle^n_0)$.
   Else, $(s_0, s_1) = (-r_0 + \langle a\rangle^n_0, -r_0)$.
3: $P_0$ & $P_1$ invoke an instance of $\binom{2}{1}$-OT$_\eta$ where $P_0$ is the sender
   with inputs $(s_0, s_1)$ and $P_1$ is the receiver with input $\langle c\rangle^B_1$. Let
   $P_1$'s output be $x_1$.
4: $P_1$ sets $t_0, t_1$ as follows: If $\langle c\rangle^B_1 = 0$, $(t_0, t_1) = (-r_1, -r_1 + \langle a\rangle^n_1)$.
   Else, $(t_0, t_1) = (-r_1 + \langle a\rangle^n_1, -r_1)$.
5: $P_0$ & $P_1$ invoke an instance of $\binom{2}{1}$-OT$_\eta$ where $P_1$ is the sender
   with inputs $(t_0, t_1)$ and $P_0$ is the receiver with input $\langle c\rangle^B_0$. Let
   $P_0$'s output be $x_0$.
6: For $b \in \{0,1\}$, $P_b$ outputs $\langle z\rangle^n_b = r_b + x_b$.
```

B2A: $\mathcal{F}_{\text{B2A}}$ 将布尔分享 $\langle c\rangle^B$ 转化为算术分享 $\langle d\rangle^n$ 满足 $d=c$。因为 $d=\langle c\rangle_0^B+\langle c\rangle_1^B-2\langle c\rangle_0^B\langle c\rangle_1^B$，则关键点在于求乘积。如算法7所示，根据 $\binom{2}{1}-\text{COT}_\eta$，$y_1=x+\langle c\rangle_0^B\langle c\rangle_1^B$。因此，$\langle d\rangle_0^n=\langle c\rangle_0^B+2x$，$\langle d\rangle_1^n=\langle c\rangle_1^B-2x-2\langle c\rangle_0^B\langle c\rangle_1^B$。由于调用了一次COT，通信量为 $\lambda+\eta$ 比特。

```
Algorithm 7 Boolean to Arithmetic, $\Pi^n_{\text{B2A}}$:

Input: $P_0, P_1$ hold $\langle c\rangle^B_0$ and $\langle c\rangle^B_1$, respectively, where $c \in \{0,1\}$.
Output: $P_0, P_1$ learn $\langle d\rangle^n_0$ and $\langle d\rangle^n_1$, respectively, s.t. $d = c$.

1: $P_0$ & $P_1$ invoke an instance of $\binom{2}{1}$-COT$_\eta$ where $P_0$ is the sender
   with correlation function $f(x) = x + \langle c\rangle^B_0$ and $P_1$ is the receiver
   with input $\langle c\rangle^B_1$. Party $P_0$ learns $x$ and sets $y_0 = n - x$ and $P_1$
   learns $y_1$.
2: For $b \in \{0,1\}$, $P_b$ computes $\langle d\rangle^n_b = \langle c\rangle^B_b - 2 \cdot y_b$.
```

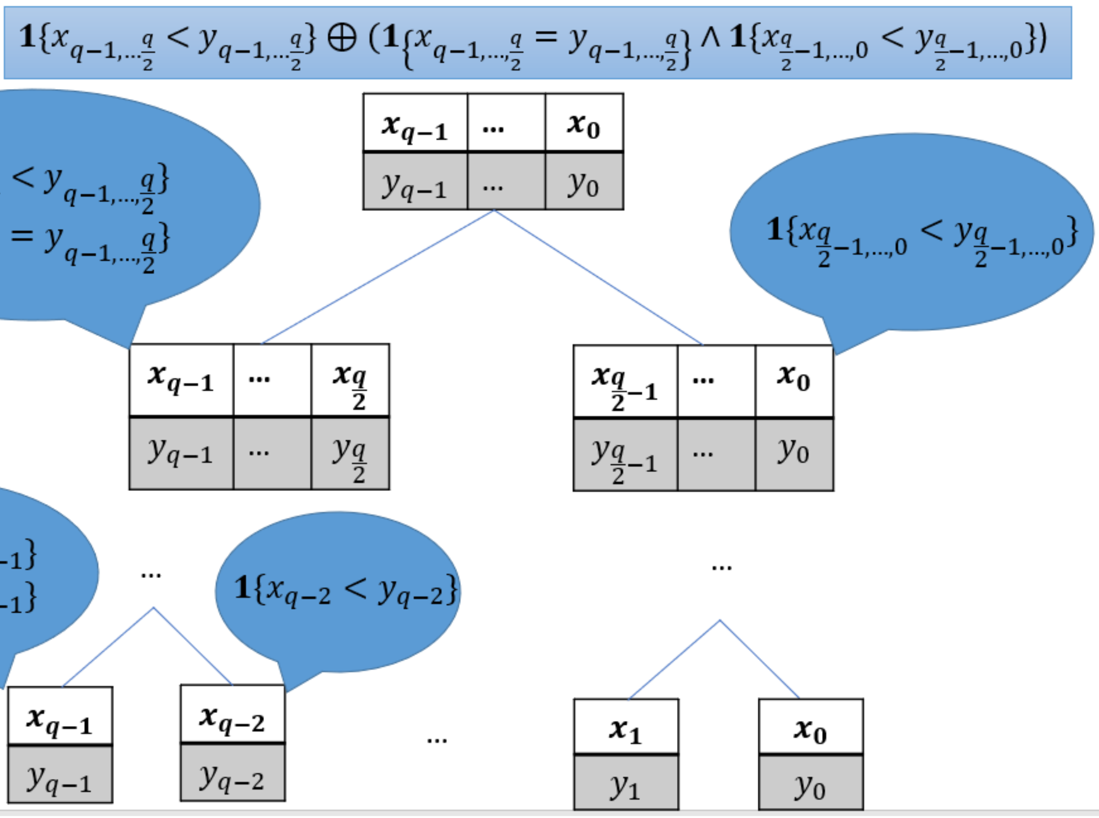
2. Protocol for Millionaires’

首先，我们介绍本文提出的解决姚氏百万富翁问题 $\mathcal{F}^{\ell}_{\text{MILL}}$: P_0 具有输入 x，P_1的输入是 y。计算 $1\{x < y\}$ 的布尔分享。其中, x 和 y 都是 ℓ 比特的无符号整数。本文的优化主要依赖如下观察:

$$1\{x < y\} = 1\{x_1 < y_1\} \oplus (1\{x_1 = y_1\} \wedge 1\{x_0 < y_0\}), \qquad (1)$$

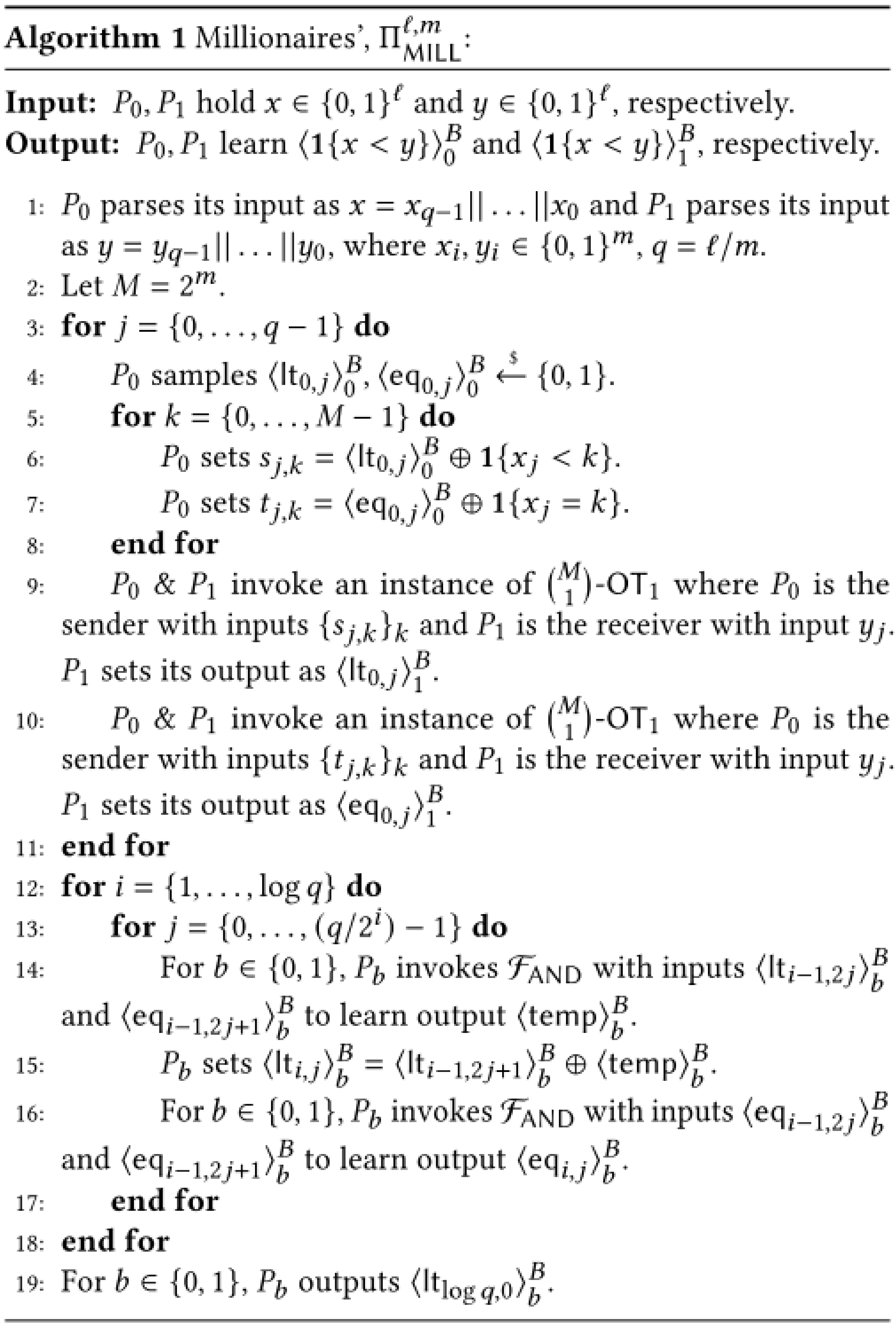
其中， $x = x_1||x_0, y = y_1||y_0$。

2.1 Intuition



$\mathbf{1}\{x_{q-1,\dots,\frac{q}{2}} < y_{q-1,\dots,\frac{q}{2}}\} \oplus (\mathbf{1}\{x_{q-1,\dots,\frac{q}{2}} = y_{q-1,\dots,\frac{q}{2}}\} \wedge \mathbf{1}\{x_{\frac{q}{2}-1,\dots,0} < y_{\frac{q}{2}-1,\dots,0}\})$
$\mathbf{1}\{x_{q-1,\dots,\frac{q}{2}} < y_{q-1,\dots,\frac{q}{2}}\}$ $\mathbf{1}\{x_{q-1,\dots,\frac{q}{2}} = y_{q-1,\dots,\frac{q}{2}}\}$
$\mathbf{1}\{x_{\frac{q}{2}-1,\dots,0} < y_{\frac{q}{2}-1,\dots,0}\}$
$\mathbf{1}\{x_{q-1} < y_{q-1}\}$ $\mathbf{1}\{x_{q-1} = y_{q-1}\}$
$\mathbf{1}\{x_{q-2} < y_{q-2}\}$
x_{q-1} ... x_0 / y_{q-1} ... y_0
x_{q-1} ... $x_{\frac{q}{2}}$ / y_{q-1} ... $y_{\frac{q}{2}}$
$x_{\frac{q}{2}-1}$ ... x_0 / $y_{\frac{q}{2}-1}$ ... y_0
x_{q-1} / y_{q-1}
x_{q-2} / y_{q-2}
x_1 / y_1
x_0 / y_0

如上图所示。令 $M = 2^m$。首先考虑比较简单的 $q = \ell/m$，并且 q 是2的次方。根据公式(1)，递归 $\log_2 q$次，得到的树有 q 个叶子节点，每个叶子节点则有 m 比特。即，$x = x_{q-1}||\dots||x_0$ 和 $y = y_{q-1}||\dots||q_0$，其中 $x_i, y_i \in \{0,1\}^m$。如此，两方则可以利用 $\binom{M}{1}-\text{OT}$计算公式(1)中的比较计算和等值计算。计算完叶子节点之后，则可以递归计算AND 和 XOR 门直到根节点，得到最后输出。算法如下:



Algorithm 1 Millionaires', $\Pi^{\ell,m}_{\mathsf{MILL}}$:

Input: P_0, P_1 hold $x \in \{0,1\}^\ell$ and $y \in \{0,1\}^\ell$, respectively.
Output: P_0, P_1 learn $\langle \mathbf{1}\{x < y\}\rangle^B_0$ and $\langle \mathbf{1}\{x < y\}\rangle^B_1$, respectively.

```
1:  $P_0$ parses its input as $x = x_{q-1}||\dots||x_0$ and $P_1$ parses its input
    as $y = y_{q-1}||\dots||y_0$, where $x_i, y_i \in \{0,1\}^m$, $q = \ell/m$.
2:  Let $M = 2^m$.
3:  for $j = \{0,\dots,q-1\}$ do
4:      $P_0$ samples $\langle \mathsf{lt}_{0,j}\rangle^B_0, \langle \mathsf{eq}_{0,j}\rangle^B_0 \xleftarrow{\$} \{0,1\}$.
5:      for $k = \{0,\dots,M-1\}$ do
6:          $P_0$ sets $s_{j,k} = \langle \mathsf{lt}_{0,j}\rangle^B_0 \oplus \mathbf{1}\{x_j < k\}$.
7:          $P_0$ sets $t_{j,k} = \langle \mathsf{eq}_{0,j}\rangle^B_0 \oplus \mathbf{1}\{x_j = k\}$.
8:      end for
9:      $P_0$ & $P_1$ invoke an instance of $\binom{M}{1}$-$\mathsf{OT}_1$ where $P_0$ is the
    sender with inputs $\{s_{j,k}\}_k$ and $P_1$ is the receiver with input $y_j$.
    $P_1$ sets its output as $\langle \mathsf{lt}_{0,j}\rangle^B_1$.
10:     $P_0$ & $P_1$ invoke an instance of $\binom{M}{1}$-$\mathsf{OT}_1$ where $P_0$ is the
    sender with inputs $\{t_{j,k}\}_k$ and $P_1$ is the receiver with input $y_j$.
    $P_1$ sets its output as $\langle \mathsf{eq}_{0,j}\rangle^B_1$.
11: end for
12: for $i = \{1,\dots,\log q\}$ do
13:     for $j = \{0,\dots,(q/2^i)-1\}$ do
14:         For $b \in \{0,1\}$, $P_b$ invokes $\mathcal{F}_{\mathsf{AND}}$ with inputs $\langle \mathsf{lt}_{i-1,2j}\rangle^B_b$
    and $\langle \mathsf{eq}_{i-1,2j+1}\rangle^B_b$ to learn output $\langle \mathsf{temp}\rangle^B_b$.
15:         $P_b$ sets $\langle \mathsf{lt}_{i,j}\rangle^B_b = \langle \mathsf{lt}_{i-1,2j+1}\rangle^B_b \oplus \langle \mathsf{temp}\rangle^B_b$.
16:         For $b \in \{0,1\}$, $P_b$ invokes $\mathcal{F}_{\mathsf{AND}}$ with inputs $\langle \mathsf{eq}_{i-1,2j}\rangle^B_b$
    and $\langle \mathsf{eq}_{i-1,2j+1}\rangle^B_b$ to learn output $\langle \mathsf{eq}_{i,j}\rangle^B_b$.
17:     end for
18: end for
19: For $b \in \{0,1\}$, $P_b$ outputs $\langle \mathsf{lt}_{\log q,0}\rangle^B_b$.
```

根据上述推导，可以很容易验证正确性。协议安全性则可以通过 $(\binom{M}{1}-\mathrm{OT}, \mathcal{F}_{\mathsf{AND}})$-hybrid model 证明。

2.2 General Case

对于一般化场景，m 不能整除 ℓ 且 $q = \lceil \ell/m \rceil$ 不是 2 的次方。做如下改进：

1. 令 $x_{q-1} \in \{0,1\}^r$，其中 $r = \ell \mod m$，如此则算法1中的step 9 & 10调用针对 x_{q-1} 和 y_{q-1} 使用 $\binom{R}{1}-\mathrm{OT}$，$R = 2^r$；

2. 进一步，由于 q 不是 2 的次方，则无法构成完美二叉树。假设 $2^\alpha < q \le 2^{\alpha+1}$，则针对 2^α 个节点构造完美二叉树，再对剩余 $q' = q - 2^\alpha$ 个节点递归构造；最终将子树根节点按照公式(1)结合。

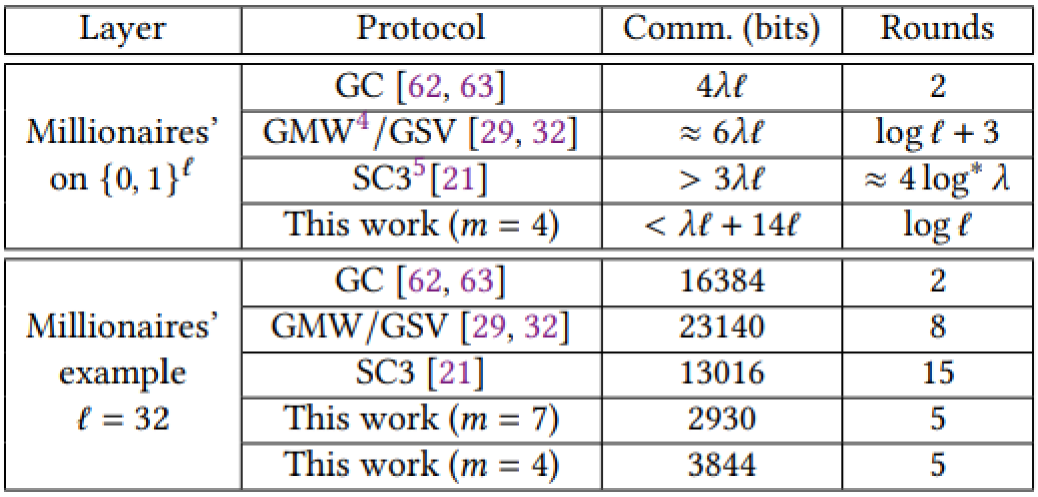
2.3 Optimizations

进一步，本文提出了如下的优化技术：

1. $2 \times \binom{M}{1}-\text{OT}_1 \Rightarrow 1 \times \binom{M}{1}-\text{OT}_2$：在算法1中的step9 & 10中，receiver的输入相同，因此发送者可以将打包发送 $\{(s_{i,k}||t_{i,k})\}_k$。则输出变为 $(\langle lt_{0,j}\rangle_1^B||\langle eq_{0,j}\rangle_1^B$。如此则将通信从 $2(2\lambda + M)$ 降低为 $2\lambda + 2M$。
2. $\mathcal{F}_{\text{AND}}$：进一步则是针对 AND 门的优化。因为在step 14 & 16 P_1 的输入相同，因此可以生成关联三元组。关联三元组可以使用 $\binom{8}{1}-\text{OT}_2$ 生成。生成关联三元组的通信量为 $2\lambda + 16$，均摊为 $\lambda + 8$。基于元组计算 AND 门则额外需要 6 比特通信。
3. Removing Unneccessary Equality: 在整个计算中 $eq_{i,0}$ ($i \in \{0, ..., \log_2 q\}$)。因此这部分计算可以省略。故而，对于计算树中最低位的分支，可以只做一个 AND 门。调用一次生成两个元组的方法生成 AND 元组，均摊通信 $\lambda + 16$。计算 AND 需要额外 4 比特。整体上，对于叶子节点节省 M 比特，对于中间节点节省 $(\lambda + 2) \cdot \lceil \log_2 q \rceil$ 比特。

2.4 Communication

在上述优化的基础上，协议需要 1次 $\binom{M}{1}-\text{OT}_1$，$q-2$ 次 $\binom{M}{1}-\text{OT}_2$，1次 $\binom{R}{1}-\text{OT}_1$，$\lceil \log_2 q \rceil$ 次 AND 和 $(q - 1 - \log_2 q)$ 次关联 AND。整体通信 $\lambda(4q - \lceil \log_2 q \rceil - 2) + M(2q - 3) + 2R + 22(q-1) - 2\lceil \log_2 q \rceil$ 比特。当 $\ell = 32$时，取 $m = 7$ 整体性能最好。



Layer	Protocol	Comm. (bits)	Rounds
Millionaires' on $\{0,1\}^\ell$	GC [62, 63]	$4\lambda\ell$	2
	GMW[4]/GSV [29, 32]	$\approx 6\lambda\ell$	$\log \ell + 3$
	SC3[5][21]	$> 3\lambda\ell$	$\approx 4 \log^* \lambda$
	This work ($m = 4$)	$< \lambda\ell + 14\ell$	$\log \ell$
Millionaires' example $\ell = 32$	GC [62, 63]	16384	2
	GMW/GSV [29, 32]	23140	8
	SC3 [21]	13016	15
	This work ($m = 7$)	2930	5
	This work ($m = 4$)	3844	5

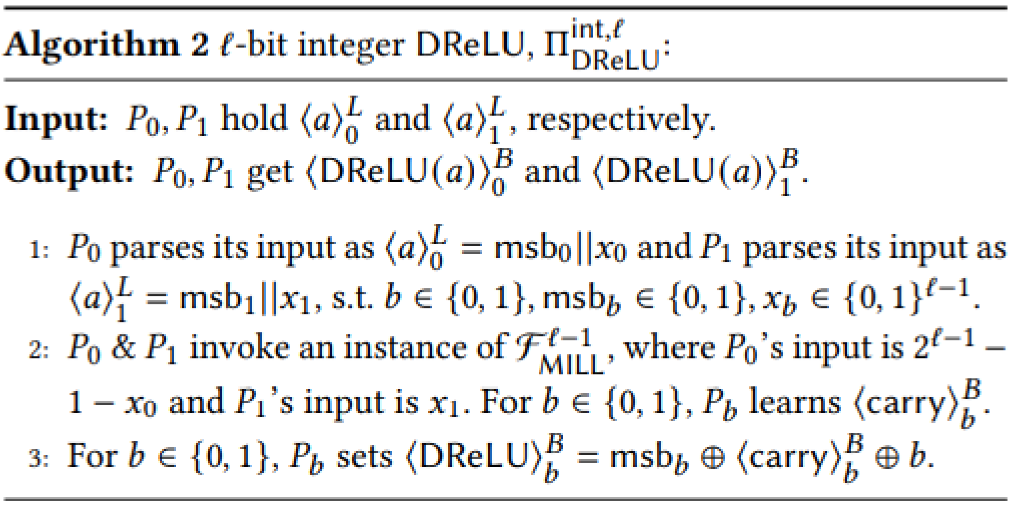
Table 1: Comparison of communication with prior work for millionaires' problem. For our protocol, m is a parameter. For concrete bits of communication we use $\lambda = 128$.

3. DReLU & ReLU

基于 $\mathcal{F}_{\text{MILL}}$ 可以构造ReLU函数。首先计算DReLU。因为 $\text{DReLU}(a) = 1 \oplus \text{MSB}(a)$，因此关键在于计算 $\text{MSB}(a)$。

3.1 DReLU in $\mathbb{Z}_L$

对于 $a \in \mathbb{Z}_L$，令 $\langle a\rangle_b^L = \mathsf{msb}_b || x_b$。则 $\text{MSB}(a) = \mathsf{msb}_0 \oplus \mathsf{msb}_1 \oplus \mathsf{carry}$，其中 $\mathsf{carry} = 1\{x_0 + x_1 > 2^{\ell-1} - 1\}$。显然 carry 可以用 $\mathcal{F}_{\text{MILL}}^{\ell-1}$ 计算得到。算法如下。正确性、安全性显然。通信则和 $\mathcal{F}_{\text{MILL}}^{\ell-1}$ 一样，简记做 $< (\lambda + 14)(\ell - 1)$ (设置 $m = 4$)。



Algorithm 2 ℓ-bit integer DReLU, $\Pi_{\text{DReLU}}^{\text{int},\ell}$:

Input: P_0, P_1 hold $\langle a\rangle_0^L$ and $\langle a\rangle_1^L$, respectively.
Output: P_0, P_1 get $\langle \text{DReLU}(a)\rangle_0^B$ and $\langle \text{DReLU}(a)\rangle_1^B$.

```
1: $P_0$ parses its input as $\langle a\rangle_0^L = \mathsf{msb}_0 || x_0$ and $P_1$ parses its input as
   $\langle a\rangle_1^L = \mathsf{msb}_1 || x_1$, s.t. $b \in \{0,1\}, \mathsf{msb}_b \in \{0,1\}, x_b \in \{0,1\}^{\ell-1}$.
2: $P_0$ & $P_1$ invoke an instance of $\mathcal{F}_{\text{MILL}}^{\ell-1}$, where $P_0$'s input is $2^{\ell-1} -$
   $1 - x_0$ and $P_1$'s input is $x_1$. For $b \in \{0,1\}$, $P_b$ learns $\langle \mathsf{carry}\rangle_b^B$.
3: For $b \in \{0,1\}$, $P_b$ sets $\langle \mathsf{DReLU}\rangle_b^B = \mathsf{msb}_b \oplus \langle \mathsf{carry}\rangle_b^B \oplus b$.
```

3.2 DReLU in $\mathbb{Z}_n$

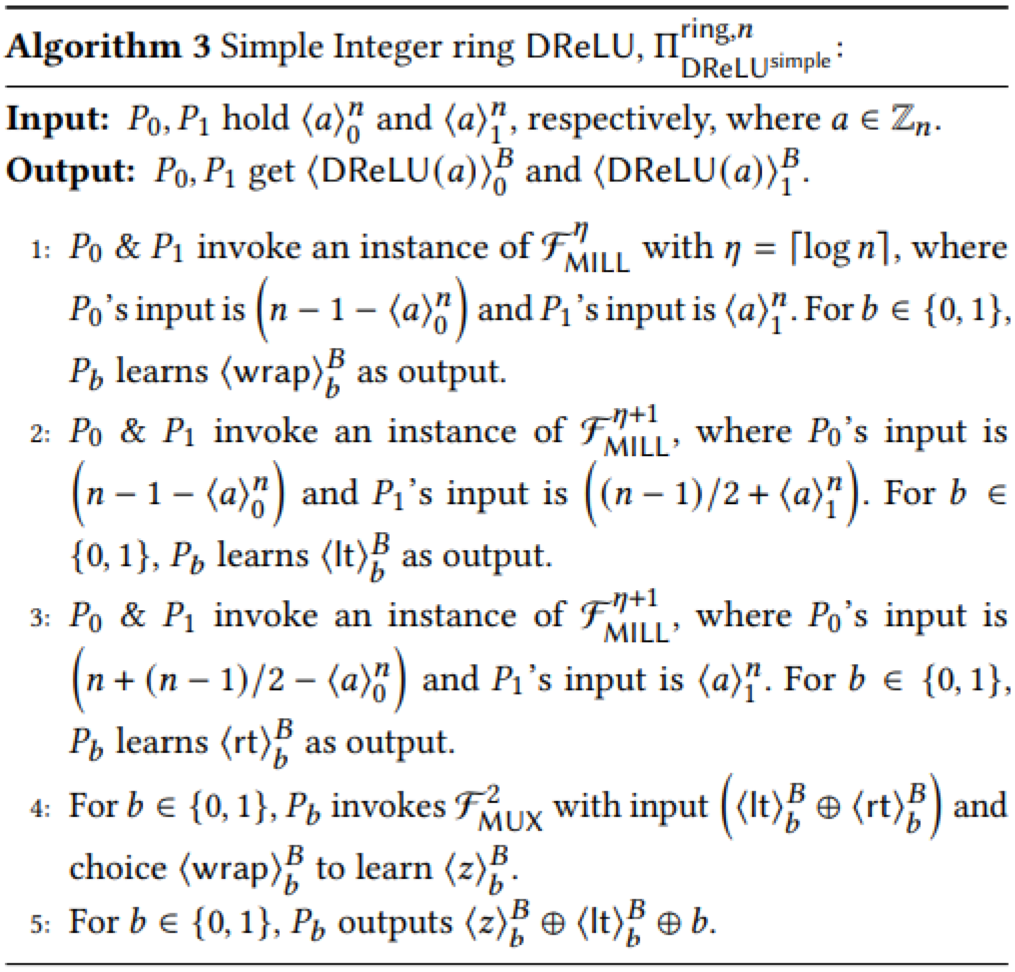
对于一般的环 $\mathbb{Z}_n$，情况稍微复杂点。其首先定义了

$$\begin{cases} \mathsf{wrap} = 1\{\langle a\rangle_0^n + \langle a\rangle_1^n > n-1\}, \\ lt = 1\{\langle a\rangle_0^n + \langle a\rangle_1^n > (n-1/2\}, \\ rt = 1\{\langle a\rangle_0^n + \langle a\rangle_1^n > n + (n-1)/2\}, \end{cases}$$

可以验证，

$$\text{DReLU}(a) = \begin{cases} 1 \oplus lt, \mathsf{wrap} = 0; \\ 1 \oplus rt, else. \end{cases}$$

如此，则得到算法如下。



Algorithm 3 Simple Integer ring DReLU, $\Pi^{\mathsf{ring},n}_{\mathsf{DReLU}^{\mathsf{simple}}}$:

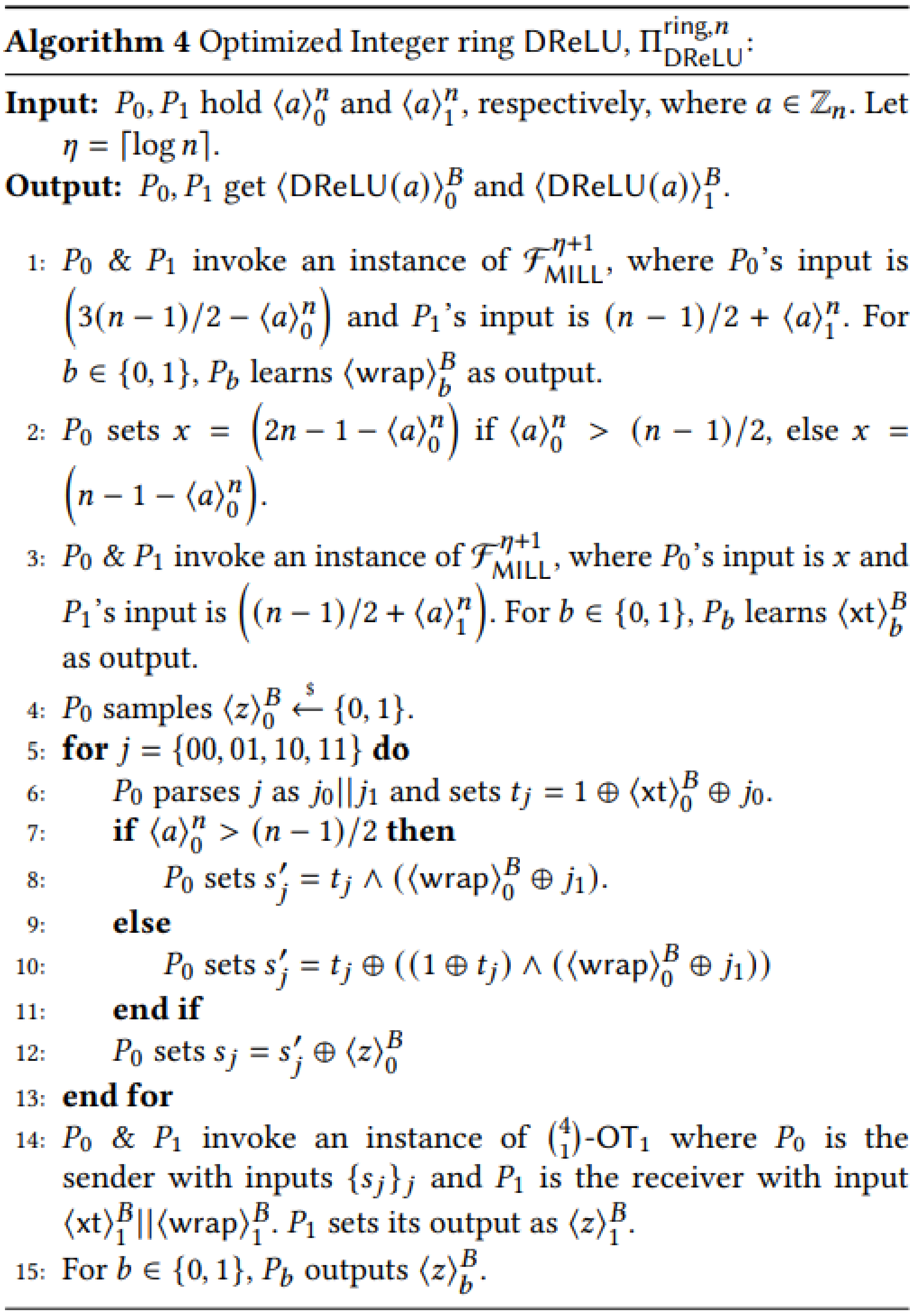
Input: P_0, P_1 hold $\langle a\rangle^n_0$ and $\langle a\rangle^n_1$, respectively, where $a \in \mathbb{Z}_n$.
Output: P_0, P_1 get $\langle \mathsf{DReLU}(a)\rangle^B_0$ and $\langle \mathsf{DReLU}(a)\rangle^B_1$.

```
1: $P_0$ & $P_1$ invoke an instance of $\mathcal{F}^{\eta}_{\mathsf{MILL}}$ with $\eta = \lceil \log n \rceil$, where $P_0$'s input is $\left(n - 1 - \langle a\rangle^n_0\right)$ and $P_1$'s input is $\langle a\rangle^n_1$. For $b \in \{0,1\}$, $P_b$ learns $\langle \mathsf{wrap}\rangle^B_b$ as output.
2: $P_0$ & $P_1$ invoke an instance of $\mathcal{F}^{\eta+1}_{\mathsf{MILL}}$, where $P_0$'s input is $\left(n - 1 - \langle a\rangle^n_0\right)$ and $P_1$'s input is $\left((n-1)/2 + \langle a\rangle^n_1\right)$. For $b \in \{0,1\}$, $P_b$ learns $\langle \mathsf{lt}\rangle^B_b$ as output.
3: $P_0$ & $P_1$ invoke an instance of $\mathcal{F}^{\eta+1}_{\mathsf{MILL}}$, where $P_0$'s input is $\left(n + (n-1)/2 - \langle a\rangle^n_0\right)$ and $P_1$'s input is $\langle a\rangle^n_1$. For $b \in \{0,1\}$, $P_b$ learns $\langle \mathsf{rt}\rangle^B_b$ as output.
4: For $b \in \{0,1\}$, $P_b$ invokes $\mathcal{F}^{2}_{\mathsf{MUX}}$ with input $\left(\langle \mathsf{lt}\rangle^B_b \oplus \langle \mathsf{rt}\rangle^B_b\right)$ and choice $\langle \mathsf{wrap}\rangle^B_b$ to learn $\langle z\rangle^B_b$.
5: For $b \in \{0,1\}$, $P_b$ outputs $\langle z\rangle^B_b \oplus \langle \mathsf{lt}\rangle^B_b \oplus b$.
```

算法3调用3次 $\mathcal{F}^{\eta}_{\text{MILL}}$ 和1次 $\mathcal{F}^{2}_{\text{MUX}}$，通信是 $3(\lambda\eta + 14\eta) + 2\lambda + 4$ 比特。

Optimization: 进一步的优化如下：

1. 首先让 P_1 将step 1，2 & 3的输入调整为一致，这样可以一起计算 $\mathcal{F}_{\text{MILL}}$ 中的叶子节点 (step 9 & 10 算法1)，具体做法时将算法3中的step 1&3 中 P_1 输入加上 $(n-1)/2$；
2. 进一步，P_0 根据自己的输入减少 step 2 或者 step 3 的执行。即如果 $\langle a\rangle^n_0 > (n-1)/2$，那么 step 2 中的 $lt = 1$；否则，step 3 中的 $rt = 0$ 恒成立。因此，可以做出如算法4中的优化。



Algorithm 4 Optimized Integer ring DReLU, $\Pi_{\mathsf{DReLU}}^{\mathsf{ring},n}$:

Input: P_0, P_1 hold $\langle a\rangle_0^n$ and $\langle a\rangle_1^n$, respectively, where $a \in \mathbb{Z}_n$. Let $\eta = \lceil \log n \rceil$.

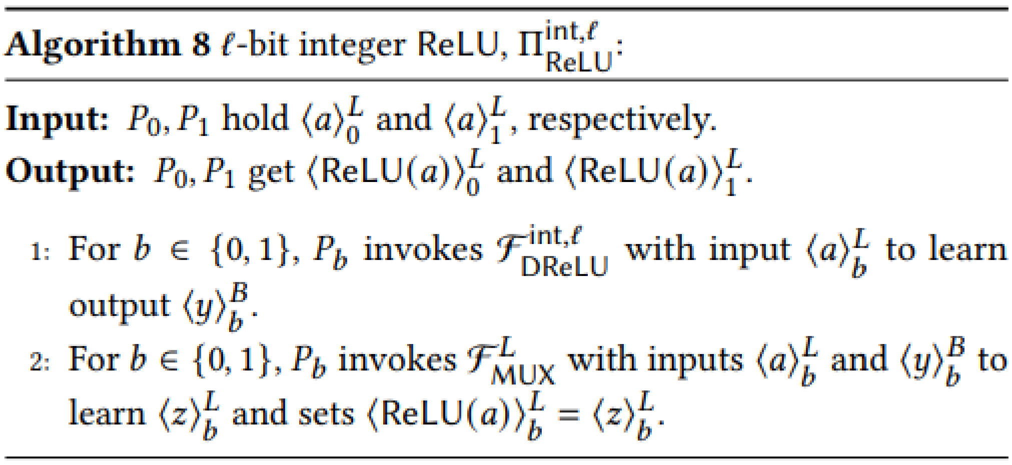
Output: P_0, P_1 get $\langle \mathsf{DReLU}(a)\rangle_0^B$ and $\langle \mathsf{DReLU}(a)\rangle_1^B$.

```
1: $P_0$ & $P_1$ invoke an instance of $\mathcal{F}_{\mathsf{MILL}}^{\eta+1}$, where $P_0$'s input is $\left(3(n-1)/2 - \langle a\rangle_0^n\right)$ and $P_1$'s input is $(n-1)/2 + \langle a\rangle_1^n$. For $b \in \{0,1\}$, $P_b$ learns $\langle \mathsf{wrap}\rangle_b^B$ as output.
2: $P_0$ sets $x = \left(2n - 1 - \langle a\rangle_0^n\right)$ if $\langle a\rangle_0^n > (n-1)/2$, else $x = \left(n - 1 - \langle a\rangle_0^n\right)$.
3: $P_0$ & $P_1$ invoke an instance of $\mathcal{F}_{\mathsf{MILL}}^{\eta+1}$, where $P_0$'s input is $x$ and $P_1$'s input is $\left((n-1)/2 + \langle a\rangle_1^n\right)$. For $b \in \{0,1\}$, $P_b$ learns $\langle \mathsf{xt}\rangle_b^B$ as output.
4: $P_0$ samples $\langle z\rangle_0^B \xleftarrow{\$} \{0,1\}$.
5: for $j = \{00, 01, 10, 11\}$ do
6:     $P_0$ parses $j$ as $j_0||j_1$ and sets $t_j = 1 \oplus \langle \mathsf{xt}\rangle_0^B \oplus j_0$.
7:     if $\langle a\rangle_0^n > (n-1)/2$ then
8:         $P_0$ sets $s'_j = t_j \wedge (\langle \mathsf{wrap}\rangle_0^B \oplus j_1)$.
9:     else
10:        $P_0$ sets $s'_j = t_j \oplus ((1 \oplus t_j) \wedge (\langle \mathsf{wrap}\rangle_0^B \oplus j_1))$
11:    end if
12:    $P_0$ sets $s_j = s'_j \oplus \langle z\rangle_0^B$
13: end for
14: $P_0$ & $P_1$ invoke an instance of $\binom{4}{1}$-OT$_1$ where $P_0$ is the sender with inputs $\{s_j\}_j$ and $P_1$ is the receiver with input $\langle \mathsf{xt}\rangle_1^B || \langle \mathsf{wrap}\rangle_1^B$. $P_1$ sets its output as $\langle z\rangle_1^B$.
15: For $b \in \{0,1\}$, $P_b$ outputs $\langle z\rangle_b^B$.
```

正确性将step14中 P_1 的输入带入到 step 5--12 中的OT构造很容易就能推导。因为 $\langle z\rangle_0^B$完全随机，所以安全性可以用 $(\mathcal{F}_{\mathsf{MILL}}^{\eta+1}, \binom{4}{1}-\mathrm{OT})$-hybrid model 证明。
算法4调用2次 $\mathcal{F}_{\mathsf{MILL}}^{\eta+1}$ (P_1输入相同) 和 一次 $\binom{4}{1}-\mathrm{OT}_1$。总通信 $< \frac{3}{2}\lambda(\eta+1) + 28(\eta+1) + 2\lambda + 4$。比算法3提升 $2\times$。

3.3 ReLU

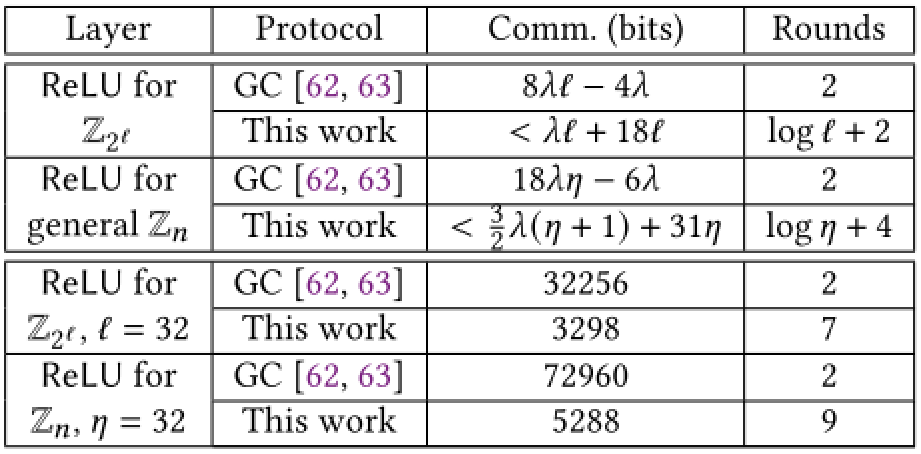
在 DReLU 的基础上，做一次 $\mathcal{F}_{\mathsf{DReLU}}$ 和 $\mathcal{F}_{\mathsf{MUX}}$。算法如下。



Algorithm 8 ℓ-bit integer ReLU, $\Pi_{\mathsf{ReLU}}^{\mathsf{int},\ell}$:

```
Input: $P_0, P_1$ hold $\langle a \rangle_0^L$ and $\langle a \rangle_1^L$, respectively.
Output: $P_0, P_1$ get $\langle \mathsf{ReLU}(a) \rangle_0^L$ and $\langle \mathsf{ReLU}(a) \rangle_1^L$.
1: For $b \in \{0,1\}$, $P_b$ invokes $\mathcal{F}_{\mathsf{DReLU}}^{\mathsf{int},\ell}$ with input $\langle a \rangle_b^L$ to learn
   output $\langle y \rangle_b^B$.
2: For $b \in \{0,1\}$, $P_b$ invokes $\mathcal{F}_{\mathsf{MUX}}^L$ with inputs $\langle a \rangle_b^L$ and $\langle y \rangle_b^B$ to
   learn $\langle z \rangle_b^L$ and sets $\langle \mathsf{ReLU}(a) \rangle_b^L = \langle z \rangle_b^L$.
```

在 $\mathbb{Z}_L$ 下，通信 $< \lambda\ell + +18\ell$；在 $\mathbb{Z}_n$ 下，通信是 $\frac{3}{2}\lambda(\eta+1)+31\eta$。单纯计算ReLU的性能如下。



Layer	Protocol	Comm. (bits)	Rounds
ReLU for $\mathbb{Z}_{2^\ell}$	GC [62, 63]	$8\lambda\ell - 4\lambda$	2
	This work	$< \lambda\ell + 18\ell$	$\log \ell + 2$
ReLU for general $\mathbb{Z}_n$	GC [62, 63]	$18\lambda\eta - 6\lambda$	2
	This work	$< \frac{3}{2}\lambda(\eta+1) + 31\eta$	$\log \eta + 4$
ReLU for $\mathbb{Z}_{2^\ell}, \ell = 32$	GC [62, 63]	32256	2
	This work	3298	7
ReLU for $\mathbb{Z}_n, \eta = 32$	GC [62, 63]	72960	2
	This work	5288	9

Table 2: Comparison of communication with garbled circuits for ReLU. We define $\eta = \lceil \log n \rceil$. For concrete bits of communication we use $\lambda = 128$.

4. Division & Truncation

本文研究的除法，是值除数公开的除法。该除法常用于神经网络计算中的Avgpool，也用于定点数乘法之后的截断。首先做理论分析，然后介绍计算除法和截断的安全协议。

4.1 Expressing general division and truncation using arithmetic over secret shares

定义 $\mathsf{idiv} : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$ 为有符号整数除法，商向 $-\infty$ 近似，余数和除数符号相同。进一步定义

$$\mathsf{rdiv}(a,d) \triangleq \mathsf{idiv}(a_u - 1\{a_u \geq \lceil n/2 \rceil\} \cdot n, d) \mod n,$$

其中 $a_u \in \{0, ..., n-1\}$ 是 $a \in \mathbb{Z}_n$ 的无符号表示，并且 $0 < d < n$ 。

定理4.1： 令 $a \in \mathbb{Z}_n$ 的秘密分享为 $\langle a\rangle_0^n, \langle a\rangle_1^n \in \mathbb{Z}_n$，$n = n^1 \cdot d + n^0$，其中 $n^0, n^1, d \in \mathbb{Z}_n$ 且 $0 \leq n^0 < d < n$。令 $a, \langle a\rangle_0^n, \langle a\rangle_1^n$ 无符号整数表示为 a_u, a_0, a_1，$a_0 = a_0^1 \cdot d + a_0^0$，$a_1 = a_1^1 \cdot d + a_1^0$，其中 $a_0^1, a_0^0, a_1^1, a_1^0 \in \mathbb{Z}$ 且 $0 \leq a_0^0, a_1^0 < d$。令 $n' = \lceil n/2 \rceil \in \mathbb{Z}$。定义 $corr$，A，B，C 如下：

$$corr = \begin{cases} -1, (a_u \geq n') \wedge (a_0 < n') \wedge (a_1 < n') \\ 1, (a_u < n') \wedge (a_0 \geq n') \wedge (a_1 \geq n') \\ 0, otherwise. \end{cases}$$
$$A = a_0^0 + a_1^0 - (1\{a_0 \geq n'\} + 1\{a_1 \geq n'\} - corr) \cdot n^0.$$
$$B = \mathsf{idiv}(a_0^0 - 1\{a_0 \geq n'\} \cdot n', d) + \mathsf{idiv}(a_1^0 - 1\{a_1 \geq n'\} \cdot n^0, d).$$
$$C = 1\{A < d\} + 1\{A < 0\} + 1\{A < -d\}.$$

从而有

$$\mathsf{rdiv}(\langle a\rangle_0^n, d) + \mathsf{rdiv}(\langle a\rangle_1^n, d) + (corr \cdot n^1 + 1 - C - B) =_n \mathsf{rdiv}(a, d).$$

证明： 对定理4.1的证明如下。首先对 $\mathsf{rdiv}(\langle a\rangle_i^n, d)$ 做分解，得到：

$$\begin{aligned} \mathsf{rdiv}(\langle a\rangle_i^n, d) &=_n \mathsf{idiv}(a_i - 1\{a_i \geq n'\} \cdot n, d) \\ &=_n \mathsf{idiv}(a_i^1 \cdot + a_i^0 - 1\{a_i \geq n'\} \cdot (n^1 \cdot d + n^0), d) \\ &=_n a_i^1 - 1\{a_i \geq n'\} \cdot n^1 + \mathsf{idiv}(a_i^0 - 1\{a_i \geq n'\} \cdot n^0, d). \end{aligned} \tag{3}$$

进一步，$a_u = a_0 + a_1 - w \cdot n$ 其中 $w = 1\{a_0 + a_1 \geq n\}$。故而有

$$\begin{aligned} a_u &= a_0 + a_1 - w \cdot n \\ &= (a_0^1 + a_1^1 - w \cdot n^1) \cdot d(a_0^0 + a_1^0 - w \cdot n^0) \\ &= (a_0^1 + a_1^1 - w \cdot n^1 + k) \cdot d(a_0^0 + a_1^0 - w \cdot n^0 - k \cdot d), \end{aligned} \tag{4}$$

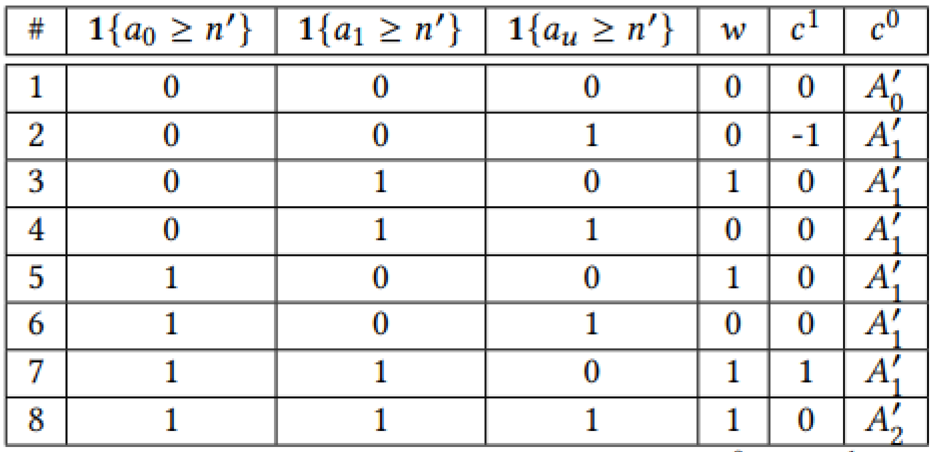
其中，$0 \leq a_0^0 + a_1^0 - w \cdot n^0 - k \cdot d < d$。
和公式(3)类似，基于公式(4)进而得到

$$\begin{aligned} \mathsf{rdiv}(a, d) &=_n a_0^1 + a_1^1 - w \cdot n^1 + k - 1\{a \geq n'\} \cdot n^1 \\ &\quad + \mathsf{idiv}(a_0^0 + a_1^0 - w \cdot n^0 - k \cdot d - 1\{a \geq n'\} \cdot n^0, d) \\ &=_n a_0^1 + a_1^1 - w \cdot n^1 - 1\{a \geq n'\} \cdot n^1 \\ &\quad + \mathsf{idiv}(a_0^0 + a_1^0 - w \cdot n^0 - 1\{a \geq n'\} \cdot n^0, d). \end{aligned} \tag{5}$$

基于公式(3)(5)，得到

$$\begin{aligned} c &=_n \mathsf{rdiv}(a,d) - \mathsf{rdiv}(\langle a\rangle_0^n,d) - \mathsf{rdiv}(\langle a\rangle_1^n,d) \\ &= (1\{a_0 \geq n'\} + 1\{a_1 \geq n'\} - w - 1\{a \geq n'\}) \cdot n^1 \\ &+ \mathsf{idiv}(a_0^0 + a_1^0 - w\cdot n^0 - 1\{a \geq n'\}\cdot n^0, d) \\ &- (\mathsf{idiv}(a_0^0 - 1\{a_0 \geq n'\}\cdot n^0, d) + \mathsf{idiv}(a_1^0 - 1\{a_1 \geq n'\}\cdot n^0, d)) \\ &=_n c^1 \cdot n^1 + c^0 - B. \end{aligned} \tag{6}$$

令 $A_i' = \mathsf{idiv}(a_0^0, a_1^0 - i\cdot n^0, d)$。$c^1$ 和 c^0 的值见下表。



#	$1\{a_0 \geq n'\}$	$1\{a_1 \geq n'\}$	$1\{a_u \geq n'\}$	w	c^1	c^0
1	0	0	0	0	0	A_0'
2	0	0	1	0	-1	A_1'
3	0	1	0	1	0	A_1'
4	0	1	1	0	0	A_1'
5	1	0	0	1	0	A_1'
6	1	0	1	0	0	A_1'
7	1	1	0	1	1	A_1'
8	1	1	1	1	0	A_2'

Table 8: Truth table for the correction terms c^0 and c^1 in the proof of division theorem (Appendix C).

通过上表，可得 $c^1 = corr$。因此 $c =_n corr\cdot n^1 + c^0 - B$。进一步，证明 $c^0 = 1 - C$。令 $C_0 = 1\{A < d\}$，$C_1 = 1\{A < 0\}$，$C_2 = 1\{A < -d\}$，那么 $C = C_0 + C_1 + C_2$。根据定理4.1中的构造和表8，我们有 $A = a_0^0 + a_1^0$ 对应第一行，$A = a_0^0 + a_1^0 - 2\cdot n^0$ 对应第八行，而其他行对应 $A = a_0^0 + a_1^0 - n^0$。因此，容易验证 $c^0 = \mathsf{idiv}(A,d)$。又容易验证 $-2\cdot d + 2 \leq A \leq 2\cdot d - 2$，因此 $c^0 \in \{-2,-1,0,1\}$:

- $c^0 = -2 \Rightarrow (A < -d) \Rightarrow (C_0 = C_1 = C_2 = 1) \Rightarrow 1 - C = -2$;
  其他情况类似。

因此，得到 $c =_n corr\cdot n^1 + (1 - C) - B$。证毕。

Corollary 4.2: 进一步，对于 ℓ 比特的整数进行截断，可以简化为

$$(a_0 \gg s) + (a_1 \gg s) + corr 2^{\ell - s} + 1\{a_0^0 + a_1^0 \geq 2^s\} =_L (a \gg s).$$

证明可以带入到定理4.1中验证。

4.2 Truncation of ℓ-bit Integer

$\mathcal{F}_{\mathsf{Trunc}}^{\mathsf{int},\ell,s}$ 对于 ℓ 比特整数，截断其末尾 s 比特。计算结果和明文计算完全相等。算法如下。

Algorithm 5 Truncation, $\Pi_{\mathsf{Trunc}}^{\mathsf{int},\ell,s}$:

```
Input: For $b \in \{0,1\}$, $P_b$ holds $\langle a\rangle_b^L$, where $a \in \mathbb{Z}_L$.
Output: For $b \in \{0,1\}$, $P_b$ learns $\langle z\rangle_b^L$ s.t. $z = a \gg s$.

1: For $b \in \{0,1\}$, let $a_b, a_b^0, a_b^1 \in \mathbb{Z}$ be as defined in Corollary 4.2.
2: For $b \in \{0,1\}$, $P_b$ invokes $\mathcal{F}_{\mathsf{DReLU}}^{\mathsf{int},\ell}$ with input $\langle a\rangle_b^L$ to learn
   output $\langle \alpha\rangle_b^B$. Party $P_b$ sets $\langle m\rangle_b^B = \langle \alpha\rangle_b^B \oplus b$.
3: For $b \in \{0,1\}$, $P_b$ sets $x_b = \mathsf{MSB}(\langle a\rangle_b^L)$.
4: $P_0$ samples $\langle \mathsf{corr}\rangle_0^L \xleftarrow{\$} \mathbb{Z}_{2^\ell}$.
5: for $j = \{00, 01, 10, 11\}$ do
6:     $P_0$ computes $t_j = (\langle m\rangle_0^B \oplus j_0 \oplus x_0) \wedge (\langle m\rangle_0^B \oplus j_0 \oplus j_1)$ s.t.
   $j = (j_0 || j_1)$.
7:     if $t_j \wedge 1\{x_0 = 0\}$ then
8:         $P_0$ sets $s_j =_L -\langle \mathsf{corr}\rangle_0^L - 1$.
9:     else if $t_j \wedge 1\{x_0 = 1\}$ then
10:        $P_0$ sets $s_j =_L -\langle \mathsf{corr}\rangle_0^L + 1$.
11:    else
12:        $P_0$ sets $s_j =_L -\langle \mathsf{corr}\rangle_0^L$.
13:    end if
14: end for
15: $P_0$ & $P_1$ invoke an instance of $\binom{4}{1}$-$\mathsf{OT}_\ell$, where $P_0$ is the sender
    with inputs $\{s_j\}_j$ and $P_1$ is the receiver with input $\langle m\rangle_1^B || x_1$
    and learns $\langle \mathsf{corr}\rangle_1^L$.
16: $P_0$ & $P_1$ invoke an instance of $\mathcal{F}_{\mathsf{MILL}}^s$ with $P_0$'s input as $2^s - 1 - a_0^0$
    and $P_1$'s input as $a_1^0$. For $b \in \{0,1\}$, $P_b$ learns $\langle c\rangle_b^B$.
17: For $b \in \{0,1\}$, $P_b$ invokes an instance of $\mathcal{F}_{\mathsf{B2A}}^L$ ($L = 2^\ell$) with
    input $\langle c\rangle_b^B$ and learns $\langle d\rangle_b^L$.
18: $P_b$ outputs $\langle z\rangle_b^L = (\langle a\rangle_b^L \gg s) + \langle \mathsf{corr}\rangle_b^L \cdot 2^{\ell-s} + \langle d\rangle_b^L$, $b \in \{0,1\}$.
```

其中step 1-15 计算 corr， step16 计算 $1\{a_0^0 + a_1^0 \geq 2^s\}$，并在 step 17 进行 B2A 转化。比较复杂的地方在于验证 $corr$ 的正确性，难点在于 step15 中 OT 的构造。将 P_1 的输入 $\langle m\rangle_1^B || x_1$ 带入，可以验证其正确性。由于 $corr$ 在计算中是完全随机的，安全性可以通过 $(\mathcal{F}_{\mathsf{DReLU}}^{\mathsf{int},\ell}, \binom{4}{1}-\mathrm{OT}, \mathcal{F}_{\mathsf{MILL}}^s, \mathcal{F}_{\mathsf{B2A}}^L)$-hybrid model 中证明。

由于调用了 $(\mathcal{F}_{\mathsf{DReLU}}^{\mathsf{int},\ell}, \binom{4}{1}-\mathrm{OT}, \mathcal{F}_{\mathsf{MILL}}^s, \mathcal{F}_{\mathsf{B2A}}^L)$ 各1次，所以通信 $\leq \lambda\ell + 2\lambda + 19\ell +$ communication for $\mathcal{F}_{\mathsf{MILL}}^s$，取决于参数 s。

4.3 Division

$\mathcal{F}_{\text{DIV}}^{\text{ring},n,d}$ 表示在一般环上的除法。该协议和截断协议很相似，不过鉴于 $-3d+2 \le A-d, A, A+d \le 3d-2$，可以用 $\delta = \lceil \log_2 6d \rceil$ 比特计算 $C = (\text{DReLU}(A-d) \oplus 1) + (\text{DReLU}(A) \oplus 1) + (\text{DReLU}(A+d) \oplus 1)$。在计算 C 之前，还需要计算 A。因此，在 $\mathbb{Z}_n$ 和 $\mathbb{Z}_\Delta(\Delta = 2^\delta)$ 上同时计算 A。算法如下。

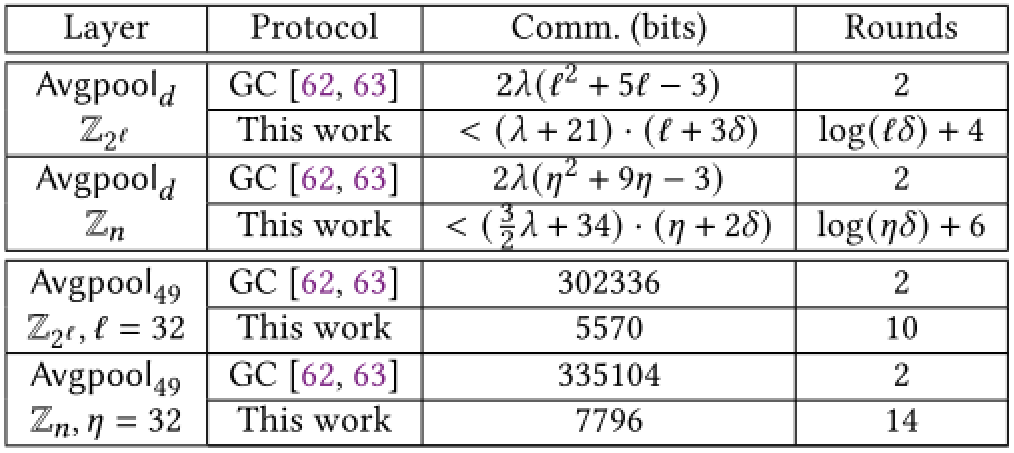
Algorithm 9 Integer ring division, $\Pi_{\text{DIV}}^{\text{ring},n,d}$:

Input: For $b \in \{0,1\}$, P_b holds $\langle a \rangle_b^n$, where $a \in \mathbb{Z}_n$.
Output: For $b \in \{0,1\}$, P_b learns $\langle z \rangle_b^n$ s.t. $z = \text{rdiv}(a,d)$.

```
1: For $b \in \{0,1\}$, let $a_b, a_b^0, a_b^1 \in \mathbb{Z}$ and $n^0, n^1, n' \in \mathbb{Z}$ be as defined in Theorem 4.1. Let $\eta = \lceil \log(n) \rceil$, $\delta = \lceil \log 6d \rceil$, and $\Delta = 2^\delta$.
2: For $b \in \{0,1\}$, $P_b$ invokes $\mathcal{F}_{\text{DReLU}}^{\text{ring},n}$ with input $\langle a \rangle_b^n$ to learn output $\langle \alpha \rangle_b^B$. Party $P_b$ sets $\langle m \rangle_b^B = \langle \alpha \rangle_b^B \oplus b$.
3: For $b \in \{0,1\}$, $P_b$ sets $x_b = \mathbf{1}\{\langle a \rangle_b^n \ge n'\}$.
4: $P_0$ samples $\langle \text{corr} \rangle_0^n \xleftarrow{\$} \mathbb{Z}_n$ and $\langle \text{corr} \rangle_0^\Delta \xleftarrow{\$} \mathbb{Z}_\Delta$.
5: for $j = \{00, 01, 10, 11\}$ do
6:     $P_0$ computes $t_j = (\langle m \rangle_0^B \oplus j_0 \oplus x_0) \wedge (\langle m \rangle_0^B \oplus j_0 \oplus j_1)$ s.t. $j = (j_0 || j_1)$.
7:     if $t_j \wedge \mathbf{1}\{x_0 = 0\}$ then
8:         $P_0$ sets $s_j =_n -\langle \text{corr} \rangle_0^n - 1$ and $r_j =_\Delta -\langle \text{corr} \rangle_0^\Delta - 1$.
9:     else if $t_j \wedge \mathbf{1}\{x_0 = 1\}$ then
10:        $P_0$ sets $s_j =_n -\langle \text{corr} \rangle_0^n + 1$ and $r_j =_\Delta -\langle \text{corr} \rangle_0^\Delta + 1$.
11:    else
12:        $P_0$ sets $s_j =_n -\langle \text{corr} \rangle_0^n$ and $r_j =_\Delta -\langle \text{corr} \rangle_0^\Delta$.
13:    end if
14: end for
15: $P_0$ & $P_1$ invoke an instance of $\binom{4}{1}$-$\text{OT}_{\eta+\delta}$ where $P_0$ is the sender with inputs $\{s_j || r_j\}_j$ and $P_1$ is the receiver with input $\langle m \rangle_1^B || x_1$. $P_1$ sets its output as $\langle \text{corr} \rangle_1^n || \langle \text{corr} \rangle_1^\Delta$.
16: For $b \in \{0,1\}$, $P_b$ sets $\langle A \rangle_b^\Delta =_\Delta a_b^0 - (x_b - \langle \text{corr} \rangle_b^\Delta) \cdot n^0$.
17: For $b \in \{0,1\}$, $P_b$ sets $\langle A_0 \rangle_b^\Delta =_\Delta \langle A \rangle_b^\Delta - b \cdot d$, $\langle A_1 \rangle_b^\Delta = \langle A \rangle_b^\Delta$, and $\langle A_2 \rangle_b^\Delta =_\Delta \langle A \rangle_b^\Delta + b \cdot d$.
18: for $j = \{0,1,2\}$ do
19:     For $b \in \{0,1\}$, $P_b$ invokes $\mathcal{F}_{\text{DReLU}}^{\text{int},\delta}$ with input $\langle A_j \rangle_b^\Delta$ to learn output $\langle \gamma_j \rangle_b^B$. Party $P_b$ sets $\langle C'_j \rangle_b^B = \langle \gamma_j \rangle_b^B \oplus b$.
20:     For $b \in \{0,1\}$, $P_b$ invokes an instance of $\mathcal{F}_{\text{B2A}}^n$ with input $\langle C'_j \rangle_b^B$ and learns $\langle C_j \rangle_b^n$.
21: end for
22: For $b \in \{0,1\}$, $P_b$ sets $\langle C \rangle_b^n = \langle C_0 \rangle_b^n + \langle C_1 \rangle_b^n + \langle C_2 \rangle_b^n$.
23: For $b \in \{0,1\}$, $P_b$ sets $B_b = \text{idiv}(a_b^0 - x_b \cdot n^0, d)$.
24: $P_b$ sets $\langle z \rangle_b^n =_n \text{rdiv}(\langle a \rangle_b^n, d) + \langle \text{corr} \rangle_b^n \cdot n^1 + b - \langle C \rangle_b^n - B_b$, for $b \in \{0,1\}$.
```

正确性验证和截断协议类似，安全性则是可以通过 $(\binom{4}{1}-\text{OT}_{\eta+\delta}, \mathcal{F}_{\text{DReLU}}^{\text{ring,n}}, \mathcal{F}_{\text{B2A}}^n)$-hybrid model.

协议调用了一次 $\mathcal{F}_{\text{DReLU}}^{\text{ring,n}}$ 和一次 $\binom{4}{1}-\text{OT}_{\eta+\delta}$，三次 $\mathcal{F}_{\text{DReLU}}^{\delta}$ 和 $\mathcal{F}_{\text{B2A}}^{n}$。总计通信 $< (\frac{3}{2}\lambda + 34) \cdot (\eta + 2\delta)$ 比特。
利用除法对Avgpool性能的提升如下：



Layer	Protocol	Comm. (bits)	Rounds
Avgpool_d $\mathbb{Z}_{2^\ell}$	GC [62, 63]	$2\lambda(\ell^2 + 5\ell - 3)$	2
	This work	$< (\lambda + 21) \cdot (\ell + 3\delta)$	$\log(\ell\delta) + 4$
Avgpool_d $\mathbb{Z}_n$	GC [62, 63]	$2\lambda(\eta^2 + 9\eta - 3)$	2
	This work	$< (\frac{3}{2}\lambda + 34) \cdot (\eta + 2\delta)$	$\log(\eta\delta) + 6$
Avgpool_{49} $\mathbb{Z}_{2^\ell}, \ell = 32$	GC [62, 63]	302336	2
	This work	5570	10
Avgpool_{49} $\mathbb{Z}_n, \eta = 32$	GC [62, 63]	335104	2
	This work	7796	14

Table 3: Comparison of communication with garbled circuits for Avgpool_d**. We define** $\eta = \lceil \log n \rceil$ **and** $\delta = \lceil \log(6 \cdot d) \rceil$**. For concrete bits of communication we use** $\lambda = 128$**. Choice of** $d = 49$ **corresponds to average pool filter of size** 7×7**.**

4.4 Truncation Optimization

对于满足 $2 \cdot n^0 \leq d = 2^s$ 的场景，$A \geq -d$ 恒成立。因此在计算 C 过程中的 $(A < -d)$ 可以省略。进一步减少 $\delta = \lceil \log_2(4d) \rceil$。

5. Secure Inference

对于神经网络模型计算，如下进行：

1. 线性层：调用基于OT的乘法或者基于HE的乘法，根据场景调节；
2. ReLU：调用ReLU协议；
3. Avgpool：调用除法协议；
4. 截断：ReLU之后输出为非负，减少开销（省略算法5的step2）；
5. Maxpool 和 Argmax：按照次序计算比较和 $\mathcal{F}_{\text{MUX}}$。

6. Evaluation

本文基于EMP实现OT，选择高效AES。同态选用SEAL。并将融入到CrypTFlow系统中。
首先和基于GC的方法比较计算 ReLU。提升 $2 - 25\times$。

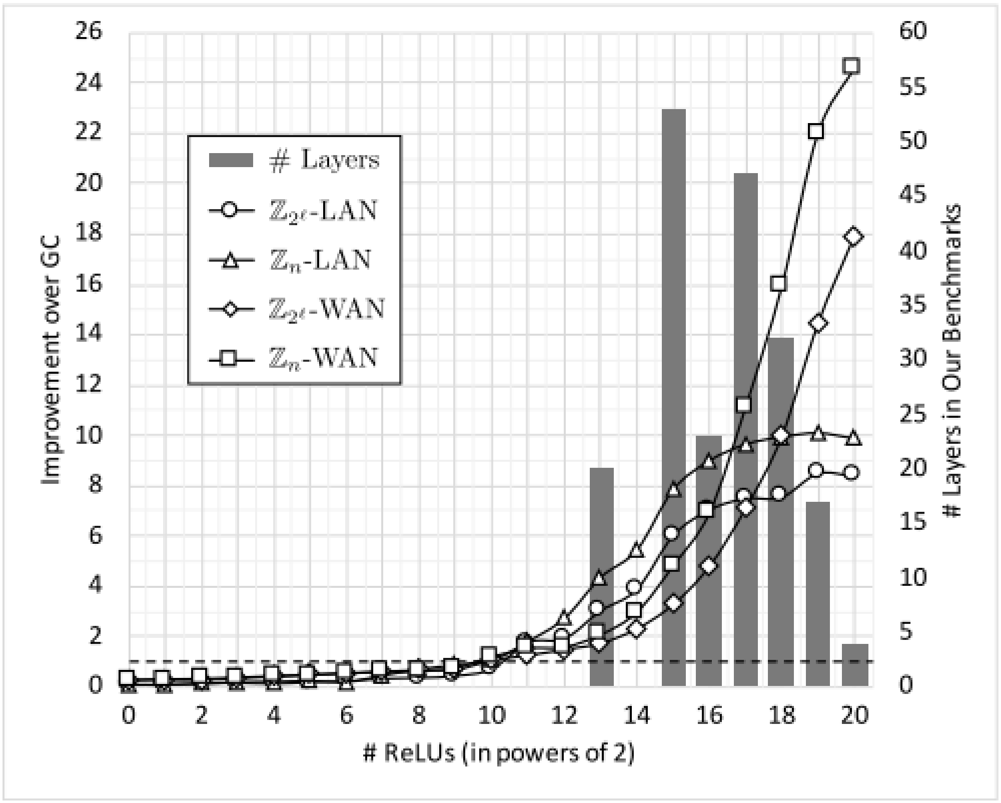


Figure 1: The left y-axis shows ($\frac{\text{GC Time}}{\text{Our Time}}$). The right y-axis shows the total number of ReLU layers corresponding to each layer size in our benchmark set. The legend entries denote the input domain and the network setting.

接下来，对于实际网络中的 ReLU 提升如下。局域网中，提升计算 $8-9\times$，广域网中提升时间 $18-21\times$，提升通信 $7-9\times$。

Benchmark	Garbled Circuits			Our Protocols		
	LAN	WAN	Comm	LAN	WAN	Comm
SqueezeNet	26.4	265.6	7.63	3.5	33.3	1.15
ResNet50	136.5	1285.2	39.19	16.4	69.4	5.23
DenseNet121	199.6	1849.3	56.57	24.8	118.7	8.21

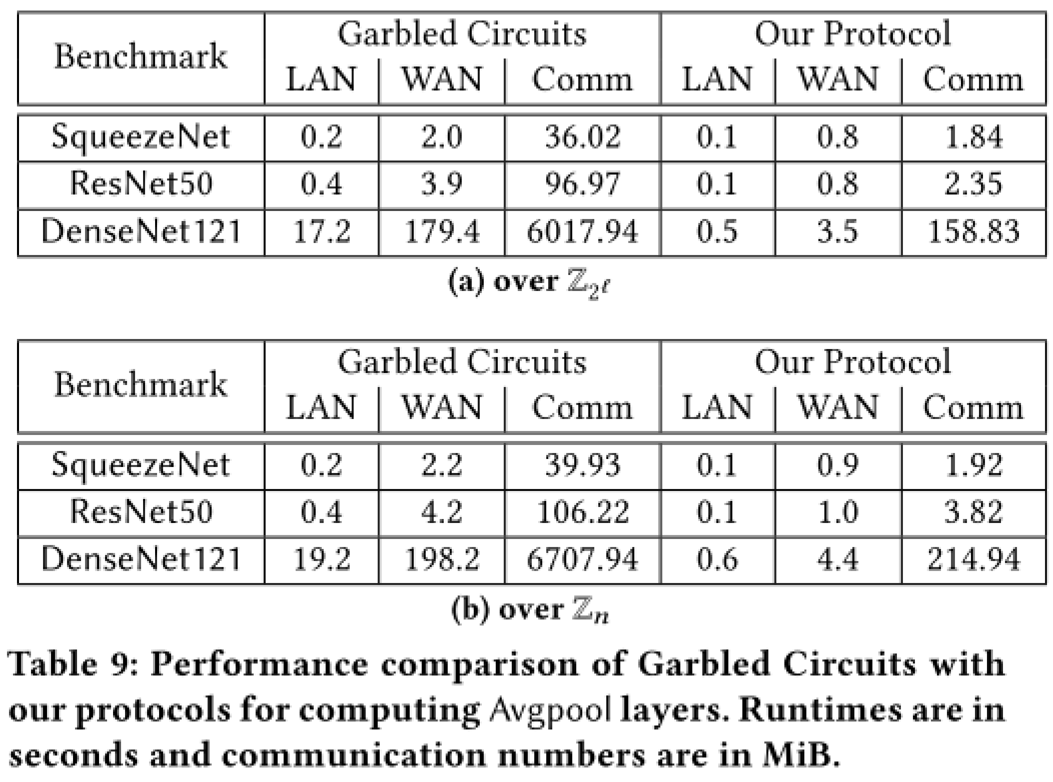
(a) over $\mathbb{Z}_{2^\ell}$

Benchmark	Garbled Circuits			Our Protocols		
	LAN	WAN	Comm	LAN	WAN	Comm
SqueezeNet	51.7	525.8	16.06	5.6	50.4	1.77
ResNet50	267.5	2589.7	84.02	28.0	124.0	8.55
DenseNet121	383.5	3686.2	118.98	41.9	256.0	12.64

(b) over $\mathbb{Z}_n$

Table 4: Performance comparison with Garbled Circuits for ReLU layers. Runtimes are in seconds and comm. in GiB.

对于 Avgpool, 提升时间 $51\times$，时间 $41\times$。



Benchmark	Garbled Circuits			Our Protocol		
	LAN	WAN	Comm	LAN	WAN	Comm
SqueezeNet	0.2	2.0	36.02	0.1	0.8	1.84
ResNet50	0.4	3.9	96.97	0.1	0.8	2.35
DenseNet121	17.2	179.4	6017.94	0.5	3.5	158.83

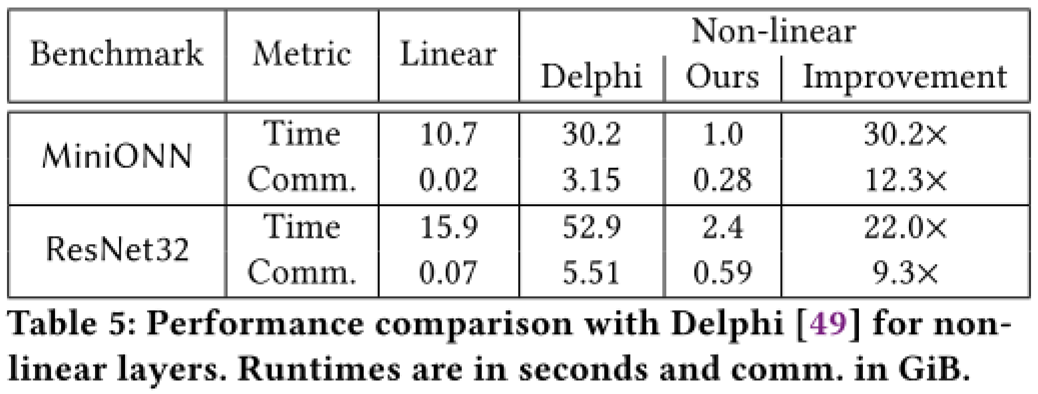
(a) over $\mathbb{Z}_{2^\ell}$

Benchmark	Garbled Circuits			Our Protocol		
	LAN	WAN	Comm	LAN	WAN	Comm
SqueezeNet	0.2	2.2	39.93	0.1	0.9	1.92
ResNet50	0.4	4.2	106.22	0.1	1.0	3.82
DenseNet121	19.2	198.2	6707.94	0.6	4.4	214.94

(b) over $\mathbb{Z}_n$

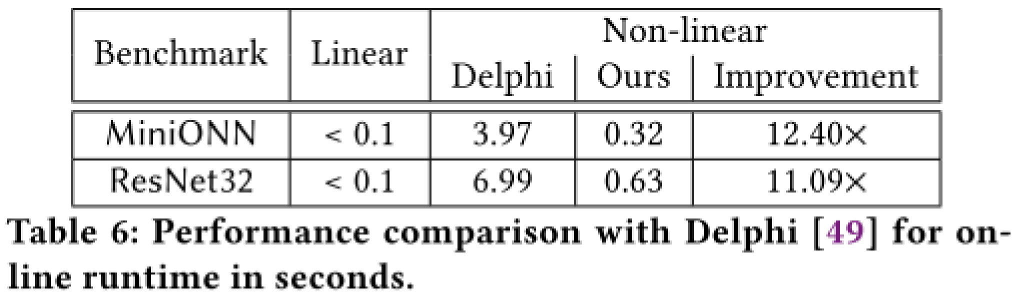
Table 9: Performance comparison of Garbled Circuits with our protocols for computing Avgpool layers. Runtimes are in seconds and communication numbers are in MiB.

和Delphi比，也仍然在非线性层和在线计算两方面都有很大提升。



Benchmark	Metric	Linear	Non-linear		
			Delphi	Ours	Improvement
MiniONN	Time	10.7	30.2	1.0	30.2×
	Comm.	0.02	3.15	0.28	12.3×
ResNet32	Time	15.9	52.9	2.4	22.0×
	Comm.	0.07	5.51	0.59	9.3×

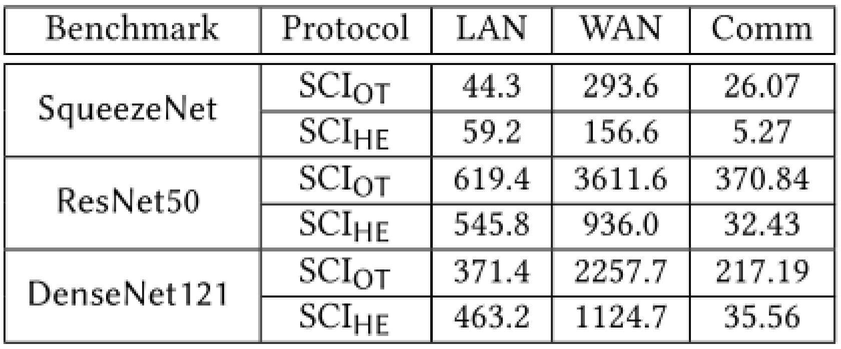
Table 5: Performance comparison with Delphi [49] for non-linear layers. Runtimes are in seconds and comm. in GiB.



Benchmark	Linear	Non-linear		
		Delphi	Ours	Improvement
MiniONN	< 0.1	3.97	0.32	12.40×
ResNet32	< 0.1	6.99	0.63	11.09×

Table 6: Performance comparison with Delphi [49] for online runtime in seconds.

最后，对于大型网络，能够在 10min (LAN) 和 20min (WAN) 内实现预测。



Benchmark	Protocol	LAN	WAN	Comm
SqueezeNet	SCI_{OT}	44.3	293.6	26.07
	SCI_{HE}	59.2	156.6	5.27
ResNet50	SCI_{OT}	619.4	3611.6	370.84
	SCI_{HE}	545.8	936.0	32.43
DenseNet121	SCI_{OT}	371.4	2257.7	217.19
	SCI_{HE}	463.2	1124.7	35.56

Table 7: Performance of CryptFlow2 on ImageNet-scale benchmarks. Runtimes are in seconds and comm. in GiB.

7. Conclusion

本文是最好的两方计算比较协议之一，是SIRNN的基础。本文提出了准确截断和除法（除数公开）的高效计算方法。这些思想对于后来的工作很有启发，有兴趣的可以继续看SIRNN的论文笔记并比较两者。