ABY^3: A Mixed Protocol Framework for Machine Learning

本次介绍的论文是Mohassel, Payman 和 Rindal, Peter发表在CCS'18上的ABY3。

ABY3

ABSTRACT

本文主要包括以下几个方面：

- 设计实现了一个通用的基于三方服务器的隐私保护机器学习框架，基于此，设计实现了新的线性回归、逻辑回归、神经网络模型
- 提出了新的完整的算术电路、布尔电路和Yao电路之间的转化协议
- 提出了新的三方秘密分享下的定点十进制小数乘法、并设计了计算分断线性多项式函数的协议
- 本文考虑了semi-honest和malicious两种情况

INTRODUCTION

本文探索基于三方服务器的隐私保护机器学习框架，改进前人的工作所遇到的挑战有以下几个方面

1. 现有的3PC技术只能适应$\mathbb{Z}_{2^k}$环上的运算，而ML中的计算或者中间参数都是以小数的形式存在，它不能直接应用模运算的代数系统上，现有的解决方案有：
   - 将小数乘上一个大因子使其成为整数，同时使用一个大的模数来避免wrap around，这样的弊端就是当计算大量浮点乘法的时候将不满足结果的正确性，同时采用了更大的模数增加了计算和能信的复杂度
   - 使用基于布尔电路和混乱电路的定点小数乘法方法，但是其效率将又大大增加
2. ML中的涉及到大量的算术计算与布尔运算，往往使用混合协议的方式比使用单一协议的方法要更高效，但是现有的转换协议已成为主要的性能瓶颈

Contributions

1. 设计实现了一种新的秘密分享下的定点数乘法协议，其吞吐量和延迟比优化的布尔电路上的实现方法分别提升了$50\times$和$24\times$。此外该协议不仅在malicious下安全而且能扩展到任意多方计算。
2. 提出新的基于三方的算术、布尔与Yao之间的转化协议，该协议能够抵抗malicious with abort的情况。
3. 提出了delayed re-share技术优化技术，在很大程度上减少向量操作中的通信复杂度，并且提出基于三方的通用扩展OT协议来计算分断多项式函数的方法。
4. 构建了semi-honest和malicious两种设定下的协议模块。
5. 实验表明比对于神经网络任务比secureML快$55000\times$，预测任务比Chameleon快270倍。

OUR FRAMEWORK

Secret Sharing

Arithmetic sharing

- 2-out-of-3 秘密分享
  1. P_1 : $(x_1, x_2), (y_1, y_2)$
  2. P_2 : $(x_2, x_3), (y_2, y_3)$
  3. P_3 : $(x_3, x_1), (y_3, y_1)$

  常见类型的计算
  - $\langle x + y \rangle := (x_1 + y_1, x_2 + y_2, x_3 + y_3)$
  - $\langle x \rangle \pm c := (x_1 \pm c, x_2, x_3)$
  - $\langle cx \rangle := (cx_1, cx_2, cx_3)$
  - 处理三方分享下的乘法先本地计算交叉项后，为保持形式的统一需要进行re-sharing的过程（将z_i发送给P_{i+1}）

$$
\begin{aligned}
\langle xy \rangle &= (x_1 + x_2 + x_3)(y_1 + y_2 + y_3) \\
&= (x_1y_1 + x_1y_2 + x_1y_3) : P_1 \to z_1 \\
&+ (x_2y_2 + x_2y_3 + x_3y_2) : P_2 \to z_2 \\
&+ (x_3y_3 + x_3y_1 + x_1y_3) : P_3 \to z_3
\end{aligned}
$$

- 优化：令$0 = a_1 + a_2 + a_3$，那么可以定义sharing为$\langle x \rangle := (x + a_1, a_2, a_3)$
- 如何生成$0 = a_1 + a_2 + a_3$
  - P_1, P_2共有一个随机数种子k_1，P_2, P_3共有一个随机数种子k_2，P_3, P_1共有一个随机数种子k_3
  - 给定一个nonce和伪随机函数 PRG
    - P_1 : $a_1 = PRG_{k_1}(nonce) - PRG_{k_2}(nonce)$;
    - P_2 : $a_2 = PRG_{k_2}(nonce) - PRG_{k_3}(nonce)$;
    - P_3 : $a_3 = PRG_{k_3}(nonce) - PRG_{k_1}(nonce)$
  - k_i可以在初始化阶段生成，$k_{i+1} = PRG(k_i)$，所以不需要通信

Binary sharing

和 Arithmetic sharing 语义类似，但是计算在模2$(\mod 2)$上。加法和乘法对应为抑或（$\oplus$）和与（$\wedge$）。

Yao's sharing

三方的Yao's sharing, 其中P_0作为Evaluator，P_1 和 P_2做Garbler。

Fixed-point Arithmetic

假设$\langle x^{'} \rangle = \langle y \rangle \langle z \rangle \in \mathbb{Z}_{2^k}$表示是$y, z$在秘文下的乘法，最后将结果以秘密分享的形式表示为$\langle x^{'} \rangle = (x^{'} + r^{'}, -r^{'})$，都是定点数表示，其正确结果需要除以$2^d$（小数部分用$d$-bit表示），即是说正确的结果应是$\langle x \rangle = \langle x^{'}/2^d \rangle$。现有的定点截断方法采用的是将$\langle \tilde{x} \rangle := (\frac{x^{'}+r^{'}}{2^d}, -\frac{r^{'}}{2^d})$，也就是说直接在秘文下进行截断，可以证明还原后其值以可忽略的概率使得误差只有最后1-bit。

这种方式存在两个问题

1. 精确性问题：1）截断直接抹掉了第d位处的进位；2）如果$\langle \tilde{x} \rangle$的分享值最高符号位与原数的最高符号位不一致（比如一个负数其秘密分享后最高符号位可能都是0，那么截断后前面$d+1$位均是0）那么会导致还原错误；
2. 针对上述问题，可以限制$|x^{'}| < 2^l \ll 2^k$，但是不能扩展到3-PC的情况。

基于三方的截断方法一Π_{trunc1}

令每方有$\langle x^{'} \rangle = \langle y \rangle \langle z \rangle$的2-out-of-3分享份额，现在的目标是计算截断$\langle x \rangle = \langle x^{'}/2^d \rangle$（转为两方的设置下来做）。

- 定义在P_1, P_2、之间的2-out-of-2秘密分享$(x_1^{'}, x_2^{'} + x_3^{'})$，然后本地截断$(x_1^{'}/2^d, (x_2^{'} + x_3^{'})/2^d)$， 最后的结果表示为$\langle x \rangle := (x_1^{'}/2^d, (x_2^{'} + x_3^{'})/2^d - r, r)$，然后做一次re-sharing的步骤。

基于三方的截断方法二Π_{trunc2}

回顾一下三方秘密分享的步骤为：1）计算$\langle x^{'} \rangle$的3-out-of-3秘密分享；然后，2）进行2-out-of-3秘密分享。Π_{trunc2}合并计算过程，从而只需要一轮通信，首先三方使用bool 电路上的减法生成$\langle r \rangle = \langle r^{'}/2^d \rangle$

- Step1秘密分享$\langle x^{'} - r^{'} \rangle$
- Step2定义$\langle x \rangle := (x^{'} - r^{'})/2^d + \langle r \rangle$，因为这里$\langle r \rangle = \langle r^{'}/2^d \rangle$

Parameters: A single 2-out-of-3 (or 3-out-of-3) share $[\![x']\!]^{\mathsf{A}} = (x'_1, x'_2, x'_3)$ over the ring $\mathbb{Z}_{2^k}$ and a integer $d < k$.
Preprocess:

1. All parties locally compute $[\![r']\!]^{\mathsf{B}} \leftarrow \mathsf{Rand}((\mathbb{Z}_2)^k)$.
2. Define the sharing $[\![r]\!]^{\mathsf{B}}$ to be the $k-d$ most significant shares of $[\![r']\!]^{\mathsf{B}}$, i.e. $r = r'/2^d$.
3. The parties compute $[\![r'_2]\!]^{\mathsf{B}}, [\![r'_3]\!]^{\mathsf{B}} \leftarrow \mathsf{Rand}((\mathbb{Z}_2)^k)$ and $[\![r_2]\!]^{\mathsf{B}}, [\![r_3]\!]^{\mathsf{B}} \leftarrow \mathsf{Rand}((\mathbb{Z}_2)^{k-d})$. r'_2, r_2 is revealed to party 1,2 and r'_3, r_3 to parties 2,3 using the RevealOne routine.
4. Using a ripple carry subtraction circuit, the parties jointly compute $[\![r'_1]\!]^{\mathsf{B}} := [\![r']\!]^{\mathsf{B}} - [\![r'_2]\!]^{\mathsf{B}} - [\![r'_3]\!]^{\mathsf{B}}$, $[\![r_1]\!]^{\mathsf{B}} := [\![r]\!]^{\mathsf{B}} - [\![r_2]\!]^{\mathsf{B}} - [\![r_3]\!]^{\mathsf{B}}$ and reveal r'_1, r_1 to parties 1,3.
5. Define the preprocessed shares as $[\![r']\!]^{\mathsf{A}} := (r'_1, r'_2, r'_3)$, $[\![r]\!]^{\mathsf{A}} := (r_1, r_2, r_3)$.

Online:

1. The parties jointly compute $[\![x'-r']\!]^{\mathsf{A}}$ and then compute $(x'-r') := \mathsf{RevealAll}([\![x - r']\!]^{\mathsf{A}})$.
2. Output $[\![x]\!]^{\mathsf{A}} := [\![r]\!]^{\mathsf{A}} + (x' - r')/2^d$.

Vectorized Multiplication

对于内积计算$\overrightarrow{x} \cdot \overrightarrow{y} := \sum_{i=1}^{n} x_i y_i$，按照原始的做法将需要调用$n$次安全乘法协议，需要$O(n)$次通信。本文做了优化，只需要$O(1)$次通信和一次trunctation-pair $(\langle r' \rangle, \langle r \rangle)$。

- $$\langle \overrightarrow{x} \rangle \cdot \langle \overrightarrow{y} \rangle := \mathrm{reveal}((\sum_{i=1}^{n} \langle x_i \rangle \langle y_i \rangle) + \langle r' \rangle)/2^d - \langle r \rangle$$

也就是，首先每方本地计算3-out-of-3的$\langle x_i \rangle \langle y_i \rangle$，然后求和，求掩码，截断，最后2-out-of-3做re-sharing。

Malicious Multiplication

Semi-honest 模型下的截断乘法如前所述。但是在malicious 模型下，需要验证正确性。本文采用了 Triple Verif. Using Another Without Opening 技术。简单来说，为了在线验证$z = xy$，预计算阶段生成三元组$c = ab$，方案如下图。关于如何保证$c = ab$，则需要使用cut-and-choose技术。

Parameters: The parties hold a triple $([\![x]\!]^A, [\![y]\!]^A, [\![z]\!]^A)$ to verify that $z = xy(\mod 2^k)$ and an addition uniformly distributed triple $([\![a]\!]^A, [\![b]\!]^A, [\![c]\!]^A)$ such that $c = ab(\mod 2^k)$.

1. Each party locally compute $[\![\rho]\!]^A := [\![x]\!]^A - [\![a]\!]^A$ and $[\![\sigma]\!]^A := [\![y]\!]^A - [\![b]\!]^A$.
2. The parties run the malicious secure open($[\![\rho]\!]^A$) and open($[\![\sigma]\!]^A$) protocols and output $\perp$ if either open protocol fails.
3. The parties run the malicious secure $\delta = \text{open}([\![z]\!]^A - [\![c]\!]^A - \sigma[\![a]\!]^A - \rho[\![b]\!]^A - \sigma\rho)$ protocol and output $\perp$ if $\delta! = 0$ or if the open protocol fails. Otherwise the parties output accept.

Figure 5: Malicious secure arithmetic multiplication protocol $\Pi_{\text{mal-arith-mult}}$.

而结合本文的截断技术2，可以给出面向定点数的malicious安全模型下的乘法如下：

Parameters: A single 2-out-of-3 (or 3-out-of-3) share $[\![x']\!]^{\mathsf{A}} = (x'_1, x'_2, x'_3)$ over the ring $\mathbb{Z}_{2^k}$ and a integer $d < k$.

Preprocess:

1. All parties locally compute $[\![r']\!]^{\mathsf{B}} \leftarrow \mathsf{Rand}((\mathbb{Z}_2)^k)$.
2. Define the sharing $[\![r]\!]^{\mathsf{B}}$ to be the $k-d$ most significant shares of $[\![r']\!]^{\mathsf{B}}$, i.e. $r = r'/2^d$.
3. The parties compute $[\![r'_2]\!]^{\mathsf{B}}, [\![r'_3]\!]^{\mathsf{B}} \leftarrow \mathsf{Rand}((\mathbb{Z}_2)^k)$ and $[\![r_2]\!]^{\mathsf{B}}, [\![r_3]\!]^{\mathsf{B}} \leftarrow \mathsf{Rand}((\mathbb{Z}_2)^{k-d})$. r'_2, r_2 is revealed to party 1,2 and r'_3, r_3 to parties 2,3 using the RevealOne routine.
4. Using a ripple carry subtraction circuit, the parties jointly compute $[\![r'_1]\!]^{\mathsf{B}} := [\![r']\!]^{\mathsf{B}} - [\![r'_2]\!]^{\mathsf{B}} - [\![r'_3]\!]^{\mathsf{B}}$, $[\![r_1]\!]^{\mathsf{B}} := [\![r]\!]^{\mathsf{B}} - [\![r_2]\!]^{\mathsf{B}} - [\![r_3]\!]^{\mathsf{B}}$ and reveal r'_1, r_1 to parties 1,3.
5. Define the preprocessed shares as $[\![r']\!]^{\mathsf{A}} := (r'_1, r'_2, r'_3)$, $[\![r]\!]^{\mathsf{A}} := (r_1, r_2, r_3)$.

Online: On input $[\![x]\!]^{\mathsf{A}}, [\![y]\!]^{\mathsf{A}}$,

1. The parties run the malicious secure multiplication protocol of [28, Protocol 4.2] where operations are performed over $\mathbb{Z}_{2^k}$. This includes:
   1. Run the semi-honest multiplication protocol [28, Section 2.2] on $[\![x]\!]^{\mathsf{A}}, [\![y]\!]^{\mathsf{A}}$ to obtain a sharing of $[\![z']\!]^{\mathsf{A}} := [\![x]\!]^{\mathsf{A}}[\![y]\!]^{\mathsf{A}}$. $\oplus$ and $\wedge$ operations are replaced with $+, *$ respectively.
   2. Before any shares are revealed, run the triple verification protocol of [28, Protocol 2.24] using $([\![x]\!]^{\mathsf{A}}, [\![y]\!]^{\mathsf{A}}, [\![z']\!]^{\mathsf{A}})$.
2. In the same round that party i sends z'_i to party $i+1$ (performed in step 1a), party i sends $(z'_i - r'_i)$ to party $i+2$.
3. Before any shares are revealed, party $i+1$ locally computes $(z'_i - r'_i)$ and runs compareview$(z'_i - r'_i)$ with party $i+2$. If they saw different values both parties send $\perp$ to all other parties and abort.
4. All parties compute $(z' - r') = \sum_{i=1}^{3}(z_i - r'_i)$.
5. Output $[\![z]\!]^{\mathsf{A}} := [\![r]\!]^{\mathsf{A}} + (z' - r')/2^d$.

Figure 6: Single round share malicious secure fixed-point multiplication protocol $\Pi_{\mathsf{mal-mult}}$.

需要注意的是，在malicious 模型下的Vectorized Multiplication 需要对每一次乘法做证明。如此，则通信开销为 $O(n)$。但是，对于Matrix Multiplication，则可以在预计算生成用于验证的矩阵三元组（scalar-mult 变为matrix-mult）。那么恶意模型下在线阶段矩阵计算的开销则和半诚实模型开销一致。

Share Conversions

Bit Decomposition $\langle x \rangle^A \rightarrow \langle \overrightarrow{x} \rangle^B$

- 基本思路是将 $\langle x \rangle^A = (x_1, x_2, x_3)$作为3PC boolean circuit的输入，进行布尔加法运算，最后的结果就是 $\langle x \rangle^A$的布尔分享即$\langle \overrightarrow{x} \rangle^B$。本文进一步基于FA和PPA做了一些电路的优化，仅需要$1 + \log k$轮通信。

Bit Extraction $\langle x\rangle^A \to \langle \overrightarrow{x}[i]\rangle^B$

- 该协议计算的是 $\langle x\rangle^A$将算术分享中的第i个位置的值转化为boolean 分享$\langle \overrightarrow{x}[i]\rangle^B$;
- 基本思路同上（相当于只需要剩下的i-bit的分享值就可以计算出来），不过只需要$O(i)$个AND门电路和$O(\log i)$通信轮数。

Bit Composition

- 使用Bit Decomposition同样的布尔电路，P_2和P_3生成x_2和x_3，并生成其Binary sharing：$\langle -x_2\rangle^B, \langle -x_3\rangle^B$。将二者输入Boolean Circuit计算$\langle x_1\rangle^B = \langle x\rangle^B + \langle -x_2 - x_3\rangle^B$，再把$\langle x_1\rangle^B$公开给$P_1$和$P_3$，因此$\langle x\rangle^A = (x_1, x_2, x_3)$。

Joint Yao Input

P_1 是 Evaluator，拥有k_X^x，而P_2, P_3作为Garbler拥有$k_X^0 \in \{0,1\}^\kappa$, 全局随机偏移$\Delta \in \{0,1\}^\kappa$满足$k_X^1 = k_X^0 \oplus \Delta$。基于本文的秘密分享方案，本文构造了原语面向将两方共有的输入转化为Yao's sharing。

- 对于P_1, P_2共有的输入x，在半诚实模型下P_2可以直接生成$\langle x\rangle^Y$并将对用sharing发送给P_1。
- 但是在恶意模型下，则需要验证$\langle x\rangle^Y$确实是关于x的Yao's sharing。在恶意模型下则需要利用P_3和承诺技术验证 1) P_2和P_3发送了相同的承诺；2）$Comm(k_X^x)$确实是关于k_X^x的承诺。为了优化通信，可以让P_2发送关于承诺的哈希值；而对于P_1, P_3共有的输入，只需要调换P_2, P_3的角色即可；3）对于P_2, P_3共有的输入，所有参与方可以利用公共随机数种子生成$k_X^x \in \{0,1\}^\kappa$，而P_2, P_3在本地计算$k_X^0 = k_X^x \oplus (x\Delta)$。

Yao to Binary

- 半诚实下使用Yao's share中key的$p_x = k_X^0[0]$（permutation bit），符合Binary share的定义，因此取$\langle x\rangle^B = (x \oplus p_x \oplus r, r, p_x)$;
- 在恶意模型下，则需要验证$x \oplus p_x \oplus r$中的$p_x = k_X^0[0]$。这个过程需要使用承诺技术。具体来说，P_1 和 P_2 选择 k_r^r, 其中 P_2 将 $k_r^0 = k_r^r \oplus (r\Delta)$ 发送给 P_3。而 P_2 和 P_3 则分别将 $C_0 = Comm(k_y^{p_x})$ 和 $C_1 = Comm(k_y^{p_x})$ 发送给 P_1，其中 $k_y^0 = k_x^0 \oplus k_r^0$。$P_1$ 发送 $k_y^{x\oplus r} = k_X^x \oplus k_r^r$ 给 P_3, P_3 验证 $k_y^{x\oplus r} \in \{k_y^0, k_y^1\}$。 并且 P_1 验证 $C_{p_x\oplus x\oplus r}$ 是 $k_y^{x\oplus r}$ 的承诺，而且 $C_0 = C_1$。最终，$\langle x\rangle^B = (x \oplus p_x \oplus r, r, p_x)$，且 P_3 可以在本地计算 $x \oplus p_x \oplus r = k_y^{x\oplus r}[0] \oplus p_r$。

Binary to Yao

- 使用Joint Yao Input直接对$\langle x\rangle^B$进行Yao's 秘密分享，并对分享在Yao sharing下做加法。

Yao to Arithmetic

- 通过Yao's circuit 3PC来计算一个加法计算出$\langle x\rangle^Y$，首先Parties 1,2采样$x_2 \leftarrow \mathbb{Z}_{2^k}$，Parties 2,3采样$x_3 \leftarrow \mathbb{Z}_{2^k}$，然后使用Yao's circuit来计算$\langle x_1\rangle^Y = \langle x\rangle^Y - \langle x_2\rangle^Y - \langle x_3\rangle^Y$。最后就得到了$\langle x\rangle^A = (x_1, x_2, x_3)$

Arithmetic to Yao

- 直接将$\langle x_i \rangle^Y$使用joint input 转化为$\langle x_i \rangle^Y$，然后在Yao's Circuit下做加法$\langle x \rangle^Y = \langle x_1 \rangle^Y + \langle x_2 \rangle^Y + \langle x_3 \rangle^Y$。

Computing $\langle a \rangle^A \langle b \rangle^B = \langle ab \rangle^A$

现在已经可以在任意的share上进行转化了，不过对于特定的计算任务仍然可以继续优化，本文提出了一种$\langle a \rangle^A \langle b \rangle^B = \langle ab \rangle^A$混合计算协议，被应用在分断线性或者多项式函数来近似非线性激活函数的计算中。在半诚实模型下，构造如下：

- 首先提出了Three-Party OT，被形式化定义为$((m_0, m_1), c, c) \to (\perp, m_c, \perp)$，分三个角色：Sender、Receiver、Helper。
  - Sender和Helper随机生成$w_0, w_1 \leftarrow \{0,1\}^k$;
  - Sender发送$m_0 \oplus w_0, m_1 \oplus w_1$给Receiver;
  - Helper发送w_c给Receiver，那么Receiver就可以得到$m_c = m_c \oplus w_c \oplus w_c$.
- 基于此，首先来计算$a\langle b \rangle^B = \langle ab \rangle^A$，其中$a \in \mathbb{Z}_{2^k}$被公开，$b \in 0,1$被Boolean分享
  - P_3（Sender）采样$r \leftarrow \mathbb{Z}_{2^k}$，定义$m_i = (i \oplus b_1 \oplus b_3)a - r, i \in 0,1$
  - P_2（Receiver）取到$m_{b2} = (b_2 \oplus b_1 \oplus b_3)a - r = ba - r$
  - P_1（Helper）已知b_2按照3PC-OT的流程将掩码发给P_2

  此时，使用zero sharing 得到(s_1, s_2, s_3)，一次resharing 后得到$\langle c \rangle^A = \langle ab \rangle^A = (s_1 + r, ab - r + s_3, s_3)$。为了减少通信轮数，可以并行进行$P_3$（Sender）和$P_1$（Receiver）之间以$P_2$为了Helper的3-OT。
- 发现上述a也可以是算术分享的形式，因此可以计算$\langle a \rangle^A \langle b \rangle^B = \langle ab \rangle^A$
  - $\langle a \rangle^A \langle b \rangle^B = a_1 \langle b \rangle^B + (a_2 + a_3)\langle b \rangle^B$
  - P_1扮演Sender计算第1项，P_3扮演Sender计算第2项即可
- 总的每方需要1轮共计4k-bit的通信。

但是上述方案并不是推广到恶意模型下。针对恶意模型的方案构造如下：

- $a\langle b \rangle^B = \langle ab \rangle^A$：首先各方本地将$\langle b \rangle^B = (b_1, b_2, b_3)$转化为$(\langle b_1 \rangle^A, \langle b_2 \rangle^A, \langle b_3 \rangle^A)$，然后在算术电路上计算XOR。具体来说先计算$\langle d \rangle^A = \langle b_1 \oplus b_2 \rangle = \langle b_1 \rangle^A + \langle b_2 \rangle^A - 2\langle b_1 \rangle^A \langle b_2 \rangle^A$，类似的进一步计算$\langle d \oplus b_3 \rangle^A$;最终，计算$\langle ab \rangle^A = a\langle b \rangle^A$。
- $\langle a \rangle^A \langle b \rangle^B = \langle ab \rangle^A$：首先将$\langle b \rangle^B$转化为$\langle b \rangle^A$，然后使用面向恶意模型的乘法协议计算$\langle ab \rangle^A$。

Polynomial Piecewise Functions

令$f_1, \ldots, f_m$表示$f(x)$的分段函数，$f(x) = f_i(x)$当$c_{i-1} < x \le c_i$时成立。因此首先判定x落在什么区间（即计算$\langle x \rangle < c$），得到$b_1, \ldots, b_m \in \{0,1\}$满足$b_i = 1 \leftrightarrow c_{i-1} < x \le c_i$，那么$f(x) = \sum_i b_i f_i(x)$。

Evaluation

本文对各种转化协议的开销做了理论分析，并在面向机器学习下的线性回归（训练和预测）、Logistic回归（训练和预测）和神经网络（仅预测）做了相关实验。

转化协议理论开销分析

Conversion	Semi-honest		Malicious	
	Comm.	Rounds	Comm.	Rounds
$[\![x]\!]^{\mathsf{A}} \to [\![x]\!]^{\mathsf{B}}$	$k + k \log k$	$1 + \log k$	$k + k \log k$	$1 + \log k$
$([\![x]\!]^{\mathsf{A}}, i) \to [\![x[i]]\!]^{\mathsf{B}}$	k	$1 + \log k$	$2k$	$1 + \log k$
$[\![x]\!]^{\mathsf{B}} \to [\![x]\!]^{\mathsf{A}}$	$k + k \log k$	$1 + \log k$	$k + \log k$	$1 + \log k$
$[\![b]\!]^{\mathsf{B}} \to [\![b]\!]^{\mathsf{A}}$	$2k$	1	$2k$	2
$[\![b]\!]^{\mathsf{Y}} \to [\![b]\!]^{\mathsf{B}}$	1/3	1	$2\kappa/3$	1
$[\![b]\!]^{\mathsf{B}} \to [\![b]\!]^{\mathsf{Y}}$	$2\kappa/3$	1	$4\kappa/3$	1
$[\![x]\!]^{\mathsf{Y}} \to [\![x]\!]^{\mathsf{A}}$	$4k\kappa/3$	1	$5k\kappa/3$	1
$[\![x]\!]^{\mathsf{A}} \to [\![x]\!]^{\mathsf{Y}}$	$4k\kappa/3$	1	$8k\kappa/3$	1

Table 1: Conversion costs between arithmetic, binary and Yao representations. Communication (Comm.) is measured in average bits per party. $x \in \mathbb{Z}_{2^k}$ is an arithmetic value, $b \in \{0, 1\}$ is a binary value, κ is the computational security parameter.

机器学习实验开销

Setting	Dimension	Protocol	Batch Size B							
			Online Throughput				Online + Offline Throughput			
			128	256	512	1024	128	256	512	1024
LAN	10	This	11764	10060	7153	5042	11574	9803	6896	4125
		[43]	7889	7206	4350	4263	47	25	11	5.4
	100	This	5171	2738	993	447	5089	2744	1091	470
		[43]	2612	755	325	281	3.7	2.0	1.1	0.6
	1000	This	406	208	104	46	377	200	100	46
		[43]	131	96	45	27	0.44	0.24	0.12	0.06
WAN	10	This	24.6	24.5	24.3	23.9	20.8	20.7	20.6	20.3
		[43]	12.4	12.4	12.4	12.4	2.4	1.6	0.88	0.50
	100	This	24.5	24.1	23.7	23.3	20.7	20.4	20.1	19.4
		[43]	12.3	12.2	11.8	11.8	0.63*	0.37*	0.19*	0.11*
	1000	This	22.2	20.2	17.5	12.6	19.3	17.9	16.5	11.6
		[43]	11.0	9.8	9.2	7.3	0.06*	0.03*	0.02*	0.01*

Figure 2: Linear Regression performance measured in iterations per second (larger = better). Dimension denotes the number of features while batch size denotes number of samples used in each iteration. WAN setting has 40ms RTT latency and 40 Mbps throughput. The preprocessing for [43] was performed either using OT or the DGK cryptosystem with the faster protocol being reported above. The * symbol denotes that the DGK protocol was performed.

Setting	Dimension	Protocol	Batch Size B							
			Online				Online + Offline			
			128	256	512	1024	128	256	512	1024
LAN	10	This	2251	2053	1666	1245	2116	1892	1441	1031
		[43]	188	101	41	25	37	20	8.6	4.4
	100	This	1867	1375	798	375	1744	1276	727	345
		[43]	183	93	46	24	3.6	1.9	1.1	0.6
	1000	This	349	184	95	42	328	177	93	41
		[43]	105	51	24	13.5	0.43	0.24	0.12	0.06
WAN	10	This	4.12	4.10	4.06	3.99	3.91	3.90	3.86	3.79
		[43]	3.10	2.28	1.58	0.99	1.4	0.94	0.56	0.33
	100	This	4.11	4.09	4.03	3.94	3.91	3.89	3.84	3.74
		[43]	3.08	2.25	1.57	0.99	0.52*	0.32*	0.17 *	0.01*
	1000	This	4.04	3.95	3.78	3.47	3.84	3.75	3.59	3.32
		[43]	3.01	2.15	1.47	0.93	0.06*	0.03*	0.02*	0.01*

Figure 4: Logistic Regression performance measured in iterations per second (larger = better). See caption of Figure 2.

Model	Protocol	Batch Size	Running Time (ms)		Comm. (MB)
			Online	Total	
Linear	This	1	0.1	3.8	0.002
		100	0.3	4.1	0.008
	SecureML [43]	1	0.2	2.6	1.6
		100	0.3	54.2	160
Logistic	This	1	0.2	4.0	0.005
		100	6.0	9.1	0.26
	SecureML [43]	1	0.7	3.8	1.6
		100	4.0	56.2	161
NN	This	1	3	8	0.5
	SecureML [43]	1	193	4823	120.5
CNN	This*	1	6	10	5.2
	Chameleon [46]	1	1360	2700	12.9
	MiniONN [40]	1	3580	9329	657.5

Figure 3: Running time and communication of privacy preserving inference (model evaluation) for linear, logistic and neural network models in the LAN setting (smaller = better). [43] was evaluated on our benchmark machine and [40, 46] are cited from [46] using a similar machine. The models are for the MNIST dataset with $D = 784$ features. NN denotes neural net with 2 fully connected hidden layers each with 128 nodes along with a 10 node output layer. CNN denotes a convolutional neural net with 2 hidden layers, see [46] details. * This work (over) approximates the cost of the convolution layers with an additional fully connected layer with 980 nodes.

总结

ABY3在三方下实现了Arithmetic，Binary，和Yao分享之间的转化，并实现了相关的机器学习计算算子。更重要的，这是比较早的几篇面向恶意敌手的隐私保护机器学习高效方案。

本篇工作由酸菜鱼和李开运合作完成。