ASTRA: High Throughput 3PC over Rings with Application to Secure Prediction

论文下载链接

0. ABSTRACT

本文提出一种考虑半诚实和恶意安全性的三方计算协议，采用offline-online的范式，能够抵抗1方静态敌手攻击。半诚实安全假设下，在环$\mathbb{Z}_{2^\ell}$中，一个乘法门的在线计算阶段仅需总通信2 ring elements，这在3PC设定中是首次实现的。在恶意安全假设下，乘法计算需要4 ring elements的总通信量，通信效率高于ABY3的5 ring elements。与此同时，恶意安全设定下也实现了Fairness（恶意方接收到output，诚实方必须获得output），且不附带多余的通信量。

作者将安全3PC协议应用到ML安全预测任务上（linear regression, linear SVM regression, logistic regression, and linear SVM classification）。将计算外包给三台不合谋的服务器，其构造满足半诚实和恶意安全的设定，其性能表现是当前最优。

1. INTRODUCTION

安全3PC计算在半诚实和恶意设定下得到了相当有关注，本文主要的目标是提升效率。基于SS的方法可以在linear门实现非交互，所以重点关注multiplication门的算法设计。一个提升效率的方向是采用offine-online两阶段的方法，offline阶段执行与input无关的操作，online阶段执行与input有关的且快速的操作。现有的3PC计算协议可以分为两类：1）支持在素域下的算术计算；2）支持半诚实下环上的算术电路计算。现有的方案各有优缺点。

Our Contribution

本文所提出的协议都拥有一个的特点是在线计算的通信比3 pair少，因此也产生了更好的在线计算性能。将贡献总结如下：

1. Secure 3PC：我们的协议在半诚实安全设定下，计算乘法时在线计算阶段通信仅需要2 elements。平均下来每方的通信量小于1 elements，这在3PC下是第一次做到的。对于恶意安全的设定下，我们的乘法计算在线阶段通信需要4 elements，是现在最好的通信效率。此外，恶意安全下也满足Fairness。
2. Secure ML prediction：我们考虑MLaaS的场景，使用辅助服务器完成ML的安全预测。我们与ABY^{3}的效率进行了比较，并提出一种新颖的应用在分类协议中的常数轮安全比较协议。
3. Implementation：对于3PC，协议基于$\mathbb{Z}_{2^l}$环实现。我们使用延迟（runtime）和在线阶段的throughput作为性能比较的benchmarks。实验表明，我们的方法效率更高。

Shared Key Setup：使用预分享随机key的方式用RPF实现3PC非交互的通信。这里的keys有

1. 每对Parties有一个公共的key。k_{01}, k_{02}, k_{12}分别表示$(P_0, P_1), (P_0, P_2), (P_1, P_2)$之间共有的key
2. 三方公有的两个key。$k_{P,1}, k_{P,2}$

2. SHARING SEMANTICS

定义本文使用的三种类型的秘密分享形式：

1. $[\cdot]$-sharing：将v以加法秘密分享的形式分发给P_1和P_2，满足$v = v_1 + v_2$
2. $[\![\cdot]\!]$-sharing：一个值v被称为$[\![\cdot]\!]$-sharing 在P_0, P_1, P_2之间，那么满足
   - 存在一个λ_v, m_v满足$v = m_v - \lambda_v$
   - P_1, P_2以明文的形式知道m_v，λ_v以$[\cdot]$-sharing的形式分享在它们之间。即$[\lambda_v]_{P_1} = \lambda_{v,1}, [\lambda_v]_{P_2} = \lambda_{v,2}$
   - P_0以明文的方式知道$\lambda_{v,1}, \lambda_{v,2}$

   我们将每一方的sharing表示为$[\![v]\!]_{P_0} = (\lambda_{v,1}, \lambda_{v,2}), [\![v]\!]_{P_1} = (m_v, \lambda_{v,1}), [\![v]\!]_{P_2} = (m_v, \lambda_{v,2})$。将$v$的$[\![\cdot]\!]$-sharing表示为$[\![v]\!] = (m_v, [\lambda])$。
3. Linearity of the secret sharing scheme：
   - 给定$x, y \in \mathbb{Z}_{2^l}$的$[\cdot]$-sharing，以及常数$c_1, c_2 \in \mathbb{Z}_{2^l}$，参与方本地计算 $[c_1 x + c_2 y]$

$$
\begin{aligned}
[c_1 x + c_2 y] &= (c_1 x_1 + c_2 y_1, c_1 x_2 + c_2 y_2) \\
&= c_1 [x] + c_2 [y]
\end{aligned}
$$

   - 同理，给定$x, y \in \mathbb{Z}_{2^l}$的-$[\![\cdot]\!]$sharing，以及常数$c_1, c_2 \in \mathbb{Z}_{2^l}$，参与方本地计算 $[\![c_1 x + c_2 y]\!]$

$$
\begin{aligned}
[\![c_1 x + c_2 y]\!] &= (c_1 m_x + c_2 m_2, c_1 [\lambda_x] + c_2 [\lambda_y]) \\
&= c_1 [\![x]\!] + c_2 [\![y]\!]
\end{aligned}
$$

   这种线性性质可以使得参与方本地计算加法和明文-密文乘法。

3. OUR 3PC PROTOCOL

3.1 3PC with semi-honest security

半诚实下的协议分为三个阶段-input-sharing, circuit-evaluation和output-reconstruction。

Input-sharing

协议$\Pi_{\mathrm{Sh}}^{s}(P_i, x)$使得指定方将$x$按$[\![\cdot]\!]$-sharing。比如$P_i, i \in 1, 2$需要分享自己的输入$x$。离线阶段，$P_0, P_i$随机采样得到$\lambda_{x,i}$，所有方随机采样得到$\lambda_{x,3-i}$。这里$P_0, P_i$是可以得到$\lambda_x = \lambda_{x,i} + \lambda_{x,3-i}$的；在线阶段，$P_i$发送$m_x = x + \lambda_x$给$P_j, j \in 1, 2$，然后$P_j$令$[\![x]\!]_{P_j} = (m_x, \lambda_{x,j})$。可以看到$P_i$没有泄露自己的隐私输入。

Circuit-evaluation

参与方按照电路拓扑顺序依次计算，对于加法门$g:(w_x,w_y,w_z)$，那么可以按照Linearity of the secret sharing scheme本地计算，见协议$\Pi_{\mathrm{Add}}(w_x,w_y,w_z)$

对于乘法门$g:(w_x,w_y,w_z)$，使用乘法协议$\Pi^s_{\mathrm{Mul}}(w_x,w_y,w_z)$计算$[\![x]\!]\times[\![y]\!]=[\![z]\!]$。基本思想与ABY2.0相同，不同的是这是三方的设置。

Correctness：令$(m_x=x+\lambda_x),(m_y=y+\lambda_y)$，有$\gamma_{xy}=\lambda_x\lambda_y$证明$\Pi^s_{\mathrm{Mul}}(w_x,w_y,w_z)$是正确的。

$$\begin{aligned}m_z&=m_xm_y-m_x\lambda_y-m_y\lambda_x+\lambda_z+\gamma_{xy}\\&=(m_x-\lambda_x)(m_y-\lambda_y)+\lambda_z\\&=xy+\lambda_z\end{aligned}$$

Output-reconstruction

重构比较简单，可以看到，每两方就可以还原输出$y=m_y-\lambda_{y,1}-\lambda_{y,2}$，将重构协议命名为$\Pi^s_{\mathrm{Res}}$。

最后将前面三个阶段的协议整合起来，命令为Π^s_{3pc}

3.2 3PC with malicious security

3PC恶意安全的方案跟半诚实的方案类似，分为三个阶段。

Input Sharing and Output Reconstruction Stages：

在协议Π^s_{Sh}中λ-shares是consistent的，因为采样的时候非交互。但是，如果P_0拥有一个数x并且想创建一个inconsistent的$[\![\cdot]\!]$-sharing，那么它会发送不一致的m_x给P_1,P_2，我们只需要让P_1,P_2交换验证$H(m_x)$即可，如果不匹配则输出Abort。记Input Sharing为Π^m_{Sh}。

对于输出也是同样采用验证的方式，不匹配则输出Abort。记Reconstruction为$\Pi^m_{\mathrm{Rec}}([\![y]\!],P)$

Circuit Evaluation Stage：

Π_{Add}是本地计算所以恶意者存在的情况下是不受影响的，所以重点考虑Π^s_{Mul}遭遇一个恶意者存在的情况。此时面临着两个问题：1）P_0作为恶意者可以分享不正确的$\gamma_{xy}\neq\lambda_x\lambda_y$；2）corrupt evaluator（$P_1\ or\ P_2$）可以在恢复$m_z$阶段作恶，发送不一致的$m_z$，所以讨论$\Pi^s_{\mathrm{Mul}}$的恶意安全性需要分为$\{P_0\}$和$\{P_1,P_2\}$两个集合来讨论，但我们的方法可以将两种情况统一起来解决。主要的思想是：检察三元组的$[\![\cdot]\!]$-sharing的product-relation。具体地，

假设P_1的m_z被不正确的构造，需要检察它的正确性，可以借助P_0：P_1可以将m_x,m_y发送给P_0，因为P_0在离线阶段知道$\lambda_x,\lambda_y,\lambda_z$，所以它能计算出$m_z$并发送给$P_1$进行校验，但这样会泄露$m_x,m_y$明

文给P_0。于是考虑加一个掩码$m_x^* = m_x + \delta_x, m_y^* = m_y + \delta_y$。$P_0$计算

$$\begin{aligned} m_z^* &= -m_x^*\lambda_y - m_y^*\lambda_x + \lambda_z + 2\gamma_{xy} \\ &= -(m_x + \delta_x)\lambda_y - (m_y + \delta_y)\lambda_x + \lambda_z + 2\gamma_{xy} \\ &= (m_z - m_z m_y) - (\delta_x\lambda_y + \delta_y\lambda_x - \gamma_{xy}) \\ &= (m_z - m_x m_y) - \chi \end{aligned}$$

这里把$m_z = m_x m_y - m_x\lambda_y - m_y\lambda_x + \lambda_z + \gamma_{xy}$代入即可得到这个结论。如果$P_0$知道$\chi$那么就可以计算$m_z^*$发送给$P_1, P_2$进行判断了。下面继续推导如果使$P_0$得到$\chi$。我们发现如果向$P_0$公开$\chi$，那么$P_0$根据自己拥有的$\lambda_x, \lambda_y, \gamma_{xy}$，以及$m_x + \delta_x, m_y + \delta_y$可以得到$\lambda_y m_x + \lambda_x m_y$。所以我们要对$\chi$加一个掩码，即$\delta_x\lambda_y + \delta_y\lambda_x + \delta_z - \gamma_{xy}$，这里$\delta_z$是一个随机数。所以可以让$P_1, P_2$随机采样$\delta_x, \delta_y, \delta_z$然后做$\chi$的$[\cdot]$-sharing并发送给$P_0$，$P_0$收到后做个Add就可以得到$\chi$。这里引入了一个新的问题，如果$P_1, P_2$中的一个是恶意者，那么要保证发送正确的$\chi$的share。

梳理一下，通过前面的描述明晰了当evaluator中的一个是恶意者的情况——检测不一致m_z的问题。并遗留下来两个问题：1）当P_0作恶时，$\gamma_{xy} \neq \lambda_x\lambda_y$；2）当$P_1$或者$P_2$作恶时，对$\chi$的不正确发送。这两个问题通过离线阶段的三元组来解决。一旦P_0接收到χ后，各方本地计算$a = \delta_x - \lambda_x, b = \delta_y - \lambda_y$以及$c = (\delta_z + \delta_x\delta_y) - \chi$的$[\![\cdot]\!]$-sharing。

观察到（a, b, c）构成一个乘法三元组

$$\begin{aligned} ab &= (\delta_x - \lambda_x)(\delta_y - \lambda_y) = \delta_x\delta_y + \lambda_x\lambda_y - \delta_x\lambda_y - \delta_y\lambda_y \\ &= (\delta_x\delta_y + \delta_z) - (\delta_x\lambda_y + \delta_y\lambda_x + \delta_z - \gamma_{xy}) \\ &= (\delta_z + \delta_x\delta_y) - \chi = c \end{aligned}$$

这里$\gamma_{xy} = \lambda_x\lambda_y$。通过验证这个乘法元组（它的检验方法同《High-Throughput Secure Three-Party Computation for Malicious Adversaries and an Honest Majority》(EUROCRYPT'17)）可以解决提到的这两个遗留问题：

（1）如果P_0作恶，$\gamma_{xy} \neq \lambda_x\lambda_y$，不妨令$\gamma_{xy} = \lambda_x\lambda_y + \Delta$，那么

$$\begin{aligned} c &= (\delta_x\delta_y + \delta_z) - (\delta_x\lambda_y + \delta_y\lambda_x + \delta_z - (\gamma_{xy} - \Delta)) \\ &= (\delta_x - \lambda_x)(\delta_y - \lambda_y) - \Delta = ab - \Delta \neq ab \end{aligned}$$

（2）考虑evaluators中的一个作恶，不妨假设是P_1作恶。在发送χ_1给P_0的时候发送一个错误值$\chi_1 + \Delta$，那么P_0恢复的值为$\chi^{'} = \chi + \Delta$

$$
\begin{aligned}
c &= (\delta_x\delta_y + \delta_z) - \chi^{'} = (\delta_x\delta_y + \delta_z) - (\chi + \Delta) \\
&= (\delta_x\delta_y + \delta_z) - (\delta_x\lambda_y + \delta_y\lambda_x + \delta_z - \gamma_{xy}) - \Delta \\
&= (\delta_x - \lambda_x)(\delta_y - \lambda_y) - \Delta = ab - \Delta \neq ab
\end{aligned}
$$

（3）加上前面讨论的一个问题：在线阶段m_z不一致，不妨假设P_1发送的$[m_z]_{P1} + \Delta$给P_2。那么诚实方P_2计算得到的是$m_z + \Delta$。这种情况下，诚实方P_0在离线阶段正确计算了$\chi = \delta_x\lambda_y + \delta_y\lambda_x + \delta_z - \gamma_{xy}$。由于在线计算阶段$P_0$得到了诚实的$m_x + \delta_x, m_y + \delta_y$以及有$\gamma_{xy} = \lambda_x\lambda_y$成立。如果它将这个结果发送给$P_1, P_2$，那么$P_2$接收到的$m_z^*$与$m_z + \Delta - m_xm_y + \delta_z$不同，所以会输出Abort。

下面是完整的恶意安全下的乘法计算协议

3.3 Achieving Fairness

我们通过一个公平的Reconstruct协议将Π_{3pc}^m的安全性从可中止提升到Fairness。为了fairly reconstruct $[\![y]\!]$，可以使用commitment。在离线阶段，$\{P_0, P_1\}$承诺它们的公共share $\lambda_{y,1}$ 发送给P_2；$\{P_0, P_2\}$同样承诺公共的share $\lambda_{y,2}$ 发送给P_1；在线阶段$\{P_1, P_2\}$承诺它们的公共值m_y发送给P_0。由诚实大多数的假设下，$(P_0, P_1), (P_0, P_2), (P_1, P_2)$中至少有1对是诚实的。当承诺不匹配的时候，向其它方广播Abort。如果没有Abort的信号出现，则按照Reconstruct的规则打开y。但是这里会有一个问题，我们缺乏一个可信任的广播信道（有可能恶意方P_0向P_1发送Abort，向P_2发送contine的信号，最终结果就会出现不一致）。为解决这个问题，可以在离线阶段$(P_0, P_1), (P_0, P_2)$分别将相应的随机数r_1, r_2的承诺（$com(r_1), com(r_2)$）发送给P_2和P_1。在线阶段P_0产生continue信号的时候则只发送continue；如果P_0发送abort，会附带关于$com(r_{3-i})$的open信息o_{3-i}给P_i，这个o_{3-i}被P_i用来向P_{3-i}证明abort确实来做P_0而非P_i自己产生（因为o_{3-i}只有P_0和P_{3-i}持有）。

4. PRIVACY PRESERVING MACHINE LEARNING

使用前面的三方协议完成隐私保护ML预测任务（外包MPC的方式，三台不合谋的服务器）

4.1 Protocols for ML

（1）**Dot Product**：给定d维向量$\vec{p}, \vec{q}$，Π_{dp}的目标是计算$\vec{p} \odot \vec{q}$的$[\![\cdot]\!]$-sharing。如果直接使用Π_{Mul}，那么它的通信复杂度与d线性相关。半诚实下，可以使用如下的方式使Π_{dp}的通信复杂度与d无关：离线阶段，P_0直接以$[\cdot]$-sharing的方式分享$\gamma_{pq} = \vec{\lambda_p} \odot \vec{\lambda_q}$，而不是对每一维分享$\lambda_{p_i}\lambda_{q_i}$。在线阶段，也不是分别重构每一维$m_{p_iq_i}$，而是$m_u$，其中$u = \vec{p} \odot \vec{q}$。同样的思路在恶意安全下的协议通信也可以被优化。

（2）**Secure Comparison**：两个算术值的比较一直是实现高效隐私保护机器学习的重要挑战。给定算术分享值$[\![u]\!], [\![v]\!]$，参与方想要计算$u < v$的值，通常的办法是计算$a = u - v$，然后判断a的最高符号位$\text{msb}(a)$，SecureML和ABY[3]采用的是GC或者parallel prefix adders。我们利用一个观察：$\text{sign}(a \cdot r) = \text{sign}(a) \oplus \text{sign}(r)$，则可以把对$a$的判断转化为对$ra$与$r$的符号位的判断。

4.2 ML Prediction Functions and Abstractions

- Linear Regression：模型M有d-维参数$\vec{w}$和一个偏置b。客户端C进行一次查询，输入为d-维的$\vec{z}$，计算流程为

$$f_{\text{linr}}((\vec{w}, b), \vec{z}) = \vec{w} \odot \vec{z} + b$$

- SVM Regression：模型M有$\{\alpha_j, y_j\}_{j=1}^{k}$, d-维支持向量$\{\vec{x}_j\}_{j=1}^{k}$和一个偏置b。客户端C进行一次查询，输入为d-维的$\vec{z}$，计算流程为

$$f_{\text{svmr}}((\{\alpha_j, y_j, \vec{x}_j\}_{j=1}^{k}, b), \vec{z}) = \sum_{j=1}^{k} \alpha_j y_j (\vec{x} \odot \vec{z}) + b$$

- Logistic Regression： M和C的输入同Linear Regression，不同的是M需要提供一个$t \in [0, 1]$，C得到

$$f_{\text{logr}}((\vec{w}, b, t), \vec{z}) = \text{sign}((\vec{w} \odot \vec{z} + b) - \ln(\frac{t}{1-t}))$$

- SVM Classification：同SVM Regression，但是C的输出有些改变

$$f_{\text{svmc}}((\{\alpha_j, y_j, \vec{x}_j\}_{j=1}^{k}, b), \vec{z}) = \text{sign}(\sum_{j=1}^{k} \alpha_j y_j (\vec{x} \odot \vec{z}) + b)$$

5. IMPLEMENTATION AND BENCHMARKING

本文在从多个维度对算法协议的效率进行了比较：延迟（计算各个参与方或者服务器在LAN和WAN下的最大运行时间，以及在LAN和WAN下的在线计算阶段的throughput）。LAN下3PC的throughput通过每秒计算的AES次数来度量，WAN下throughput通过每少计算的AND门数量来比较。（只放少量的实验截图）

再看各个输入、输出阶段的实验比较

6. Conclusion

本文提出了在3PC下的恶意方案，对比ABY3在线性能提升了。并且实现了Fairness安全性。接下来的BLAZE工作也是在本文的基础上的提升。

与李开运合作完成。