在之前的博客中，我们总结梳理了基于密码学技术的安全聚合三条技术路线：基于成对加性掩码、基于同态加密和基于秘密分享。其中，以秘密分享为基础的安全聚合可以在服务器端调用安全多方计算协议，理论上可以支持任意聚合方案的安全实现。本文我们进一步深入分析该技术路线下的关键工作。

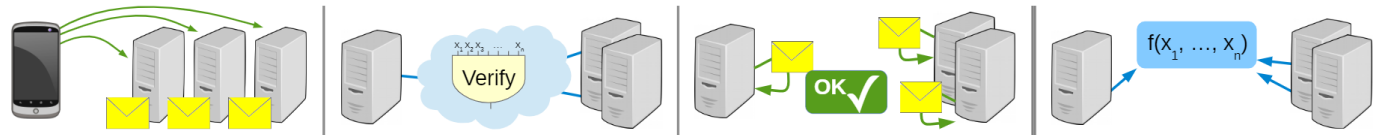
0. 系统模型概览

在介绍具体的方案构造之前，本文首先介绍该技术路线遵循的系统模型。具体来说，该方案遵循客户端-服务器架构，其中客户端有n个，服务器数量$s \geq 2$（具体数目s根据安全多方计算协议设置）。在每一轮聚合的工作流程如下：

- 每一个客户端C_i首先调用秘密分享将自己的私有输入x_i分享为多份$[x_i]_{j=1}^{s}$，每一份$[x_i]_j$发送给对应的服务器S_j；
- 服务器S_j在接收到秘密份额之后，调用安全多方计算协议计算$[x] = F([x_1], [x_2], \ldots, [x_n])$；
- 服务器S_j返回$[x]_j$给每一个客户端，客户端恢复聚合结果。

常见的聚合函数F有很多，例如联邦学习中的加法聚合等。

1. Prio: Private, Robust, and Scalable Computation of Aggregate Statistics (USENIX NSDI'17)



(a) The client sends a share of its encoded submission and SNIP proof to each server.

(b) The servers validate the client's SNIP proof to ensure that the submission is valid.

(c) If the checks pass, the servers update their local accumulators with the client-provided data.

(d) After accumulating many packets, the servers publish their accumulators to reveal the aggregate.

Figure 1: An overview of the Prio pipeline for processing client submissions.

Henry Corrigan-Gibbs等人首先提出了Prio方案，实现安全、鲁棒且可扩展的聚合统计。其系统架构如图所示，主要贡献包括三个方面：

- 提出了秘密分享下的非交互式证明（secret-shared non-interactive proofs, SNIPs），该证明是针对客户端-服务器架构的专用优化信息论零知识证明；
- 提出了仿射聚合编码（affine-aggregatable encodings），该技术利用之前的编码方案优化安全聚合下的效率；
- 最后，Prio将仿射聚合编码和SNIPs结合，同时支持安全聚合下的鲁棒和隐私两种特性。

1.1. Secret-shared Non-Interactive Proofs

接下来首先介绍SNIPs协议的构造。大致来说，SNIPs协议包含一个客户端（证明者，prover）和多个服务器（验证者，verifiers）。客户端的目标是令服务器确认验证电路$\mathrm{Valid}(x) = 1$，同时不泄露给服务器任何其他的信息。为了达到这个目标，客户端需要发送给服务器证明串（proof strings），服务器之间利用证明串、调用安全计算协议验证。该验证需要满足零知识证明的三条属性：Correctness, Soundness, Zero knowledge。

1.1.1 SNIPs构造

Set-up. 令M表示在认证电路Valid中的乘法门个数，本文工作的底层编码域记作$\mathbb{F}$，其中$2M \ll |\mathbb{F}|$。

Step 1: Client evaluation. 对于客户端来说，其知道验证电路Valid的构造和自身私有输入x，因此，其可以计算得到$\mathrm{Valid}(x)$每条输入输出线上的中间值。按照电路的拓扑结构，不失一般性，记第t个乘法门的输入为u_t和v_t。进一步，客户端利用拉格朗日插值算法计算得到多项式f和g，使得对于$1 \leq t \leq M$均有$f(t) = u_t$且$g(t) = v_t$。进一步，定义$h = f \cdot g$。如此，f和g的次数$\leq M - 1$，h的次数$\leq 2M - 2$。最后，客户端将h进行秘密分享（即将h的系数进行秘密分享），并把份额$[h]_i$发送给第i个服务器。

Step 2: Consistency chencking at the servers. 第i个服务器在接收到客户端上传的$[x]_i$和$[h]_i$之后，各个服务器可以在本地计算得到$[f]_i$和$[g]_i$。具体来说，服务器i可以进行如下操作：

- 其本身具有输入的份额$[x]_i$；
- 其拥有每一个乘法门的输出值秘密分享份额，例如对于第t个乘法门，其具有$[h]_i[t]$；
- 因此，服务器i可以在本地根据仿射操作得到所有其余电路线对应的秘密份额。

有了上述秘密份额，每一个服务器可以在本地利用多项式插值计算得到$[f]_i$和$[g]_i$。如果客户端和服务器均诚实执行上述协议，那么有$f \cdot g = h$。否则，一个恶意的客户端可能会发送$\widehat{h}$的份额给服务器，其中$\exists t \in \{1, \dots, M\}$使得$\widehat{h}(t)$不是验证电路$\mathrm{Valid}(x)$中第$t$个乘法门的输出值。那么必然会有得到的$\hat{f}$和$\hat{g}$满足$\hat{f} \cdot \hat{g} \neq \widehat{h}$。理由如下：如果存在$t_0$满足$\widehat{h}(t_0) \neq h(t_0)$，那么对于$t \leq t_0$有$\hat{f}(t) = f(t)$且$\hat{g}(t) = g(t)$。由于

$$\widehat{h}(t_0) \neq h(t_0) = f(t_0) \cdot g(t_0) = \hat{f}(t_0) \cdot \hat{g}(t_0)$$

则有$\widehat{h}(t_0) \neq \hat{f}(t_0) \cdot \hat{g}(t_0)$，因此$\widehat{h} \neq \hat{f} \cdot \hat{g}$。

Step 3a：Polynomial identity test. 为了验证$\hat{f} \cdot \hat{g} = \widehat{h}$，本文使用了Schwartz-Zippel随机多项式测试算法。该算法的大致思想如下：如果$\hat{f} \cdot \hat{g} \neq \widehat{h}$，那么$\hat{f} \cdot \hat{g} - \widehat{h}$是一个次数最多为$2M-2$的非零多项式，该多项式最多在$\mathbb{F}$中有$2M-2$个根。因此，服务器可以随机选取$r \in \mathbb{F}$然后计算$\hat{f}(r) \cdot \hat{g}(r) - \widehat{h}(r)$，那么服务器验证得到$\hat{f} \cdot \widehat{h} \neq \widehat{h}$的概率为$1 - \frac{2M-2}{|\mathbb{F}|}$。

Step 3b: Multiplication of shares. 对于3a中$[\hat{f}(r) \cdot \hat{g}(r) - \widehat{h}(r)]$的计算，关键在于计算$[\hat{f}(r) \cdot \hat{g}(r)]$。传统的MPC方案是利用Beaver乘法三元组实现，但是实现该方法需要在预计算调用同态加密或者不经意传输生成三元组，该生成开销较大。在方案Prio中，由于服务器是诚实的且不和客户端合谋，那么可以令客户端在本地生成三元组$(a, b, c) \in \mathbb{F}^3$，然后将$([a]_i, [b]_i, [c]_i)$发送给第$i$个服务器。即便客户端时恶意的，3a中检测失败的概率也仅有$\frac{2M-2}{|\mathbb{F}|}$。

Step 4： Output verification. 最后，所有的服务器公开电路$\mathrm{Valid}(x)$输出线的秘密分享$([w_1]_i, [w_2]_i, \dots)$，然后验证结果$\mathrm{Valid}(x) \stackrel{?}{=} 1$。

1.1.2 安全性和效率

SNIPs失败的概率为$\frac{2M-2}{|\mathbb{F}|}$，取$|\mathbb{F}|^{128}$，上述概率可以做到可忽略。进一步，Prio比较了SNIP和之前零知识证明方案NIZK和SNARK的性能，具体如下表：

		NIZK	SNARK	**Prio** (SNIP)
Client	Exps.	M	M	0
	Muls.	0	$M \log M$	$M \log M$
	Proof len.	M	1	M
Servers	Exps./Pairs.	M	1	0
	Muls.	0	M	$M \log M$
	Data transfer	M	1	1

Table 2: An asymptotic comparison of Prio with standard zero-knowledge techniques showing that Prio reduces the computational burden for clients and servers. The client holds a vector $x \in \mathbb{F}^M$, each server i holds a share $[x]_i$, and the client convinces the servers that each component of x is a $0/1$ value in $\mathbb{F}$. We suppress the $\Theta(\cdot)$ notation for readability.

1.2 Others

除了上述SNIPs，Prio还提出了仿射可聚合编码（Affine-aggregatable encodings，AFEs），从而将一些非线性聚合通过编码转化为编码空间内的线性聚合，提升整体性能。Prio中支持的聚合函数包括：求和与平均、方差、布尔或与、最大最小、频数统计、集合运算和机器学习中的最小平方拟合。有些运算为了提升性能需要有一定的信息泄露，Prio中对相关泄露进行了详细刻画。相关部分的具体实现和实验请参考原文。

2. Prio+：Privacy Preserving Aggregate Statistics via Boolean Shares

在Prio的基础上，Prio+为了进一步限制恶意客户端的作恶能力，强制令客户端通过布尔分享而不是算术分享上传输入。客户端在接收到布尔分享之后，则通过布尔->算术安全转化协议将布尔分享转化为算术分享，进而实现算术分享下的一些聚合计算，从而避免使用Prio中的零知识证明。该方案的核心在于布尔->算术转化，Prio+基于安全两方计算，利用daBits等技术实现了高效的在线转化。

3. ELSA：Secure Aggregation for Federated Learning with Malicious Actors (IEEE S&P'23)

在S&P'23上，Rathee等人面向联邦学习中的恶意客户端，提出了ELSA兼顾鲁棒性和数据隐私保护。该方案使用基于L2范数的鲁棒性检测算法，并部署两台服务器实现安全聚合。系统架构图如下所示：

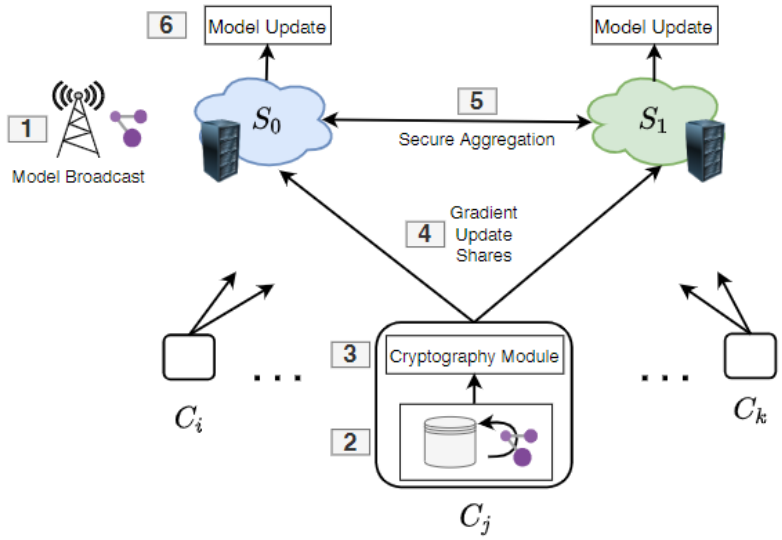


Figure 1: ELSA FL pipeline. S_0, S_1 denotes servers in different trust domains who each hold a copy of the current global model. C_i represents the ith client who performs local training before using its cryptography module to enable secure aggregation. Numbers in boxes represent steps.

ELSA方案的设计思路比较简洁：根据L2范数的鲁棒性检测算法，ELSA需要如下安全两方计算基础模块：算术分享与布尔分享、OT协议和Beaver乘法三元组。利用Beaver乘法三元组用于L2范数计算中的安全求平方运算。

3.1 直观的方案

最直观的方法则是利用两方算术秘密分享把客户端梯度直接秘密分享给两个服务器，两个服务器调用上述安全计算模块实现安全鲁棒性检测、聚合。然而，安全两方计算的离线计算的开销太大，会影响系统的整体性能。

3.2 ELSA的优化

为了减少离线开销，ELSA的优化策略和Prio中优化Beaver乘法三元组的策略很类似：让客户端去生成预计算关联随机数。然而，客户端是不可靠的：在Prio中，客户端生成的随机数即使不正确，在计算验证电路的时候也会被以绝对概率检测出来；但是在ELSA中，关联随机数要用来计算实际任务电路，没有Prio中类似的性质，所以需要保证随机数的正确性。需要验证的关联随机数包括：

- 基础OT生成的随机数；
- 平方关联随机数；

本文用的验证策略大致是令两个服务器对关联随机数进行随机线性组合+Cut&Choose策略，然后验证。该验证比简单的令两方生成关联随机数要高效很多。

3.3 其他内容

ELSA进一步还实现了恶意模型下的隐私性、减少证明串长度、客户端每轮聚合只上传一次数据等优化，相关内容可以参考原文。

4. 其他类似工作

除了上述工作之外，我们课题组的EaSTFLy和FLOD系列工作、国外的FLGUARD和HyFL等工作也采取了类似的安全聚合技术路线。