计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
2025,34(3):14–26 [doi: 10.15888/j.cnki.csa.009792] [CSTR: 32024.14.csa.009792]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

考虑外部敌手的去中心化联邦学习梯度聚合协议①

邹洁丽, 张子华, 高铁杠

(南开大学 软件学院, 天津 300350)
通信作者: 高铁杠, E-mail: gaotiegang@nankai.edu.cn

摘　要: 联邦学习是一种分布式机器学习技术, 允许参与方在本地训练模型并上传更新至中央服务器, 由中央服务器聚合更新来生成更优的全局模型, 从而保障数据隐私和解决数据孤岛问题. 然而, 梯度聚合过程依赖中央服务器, 这可能导致单点故障, 且中央服务器也是潜在的恶意攻击者. 因此, 联邦学习必须去中心化. 现有的去中心化方案没有考虑外部敌手和数据通信带来的性能瓶颈问题. 为了解决上述问题, 提出一种考虑外部敌手的去中心化联邦学习方法. 该方法应用 Shamir 秘密共享方案, 将模型更新分成多个份额, 保护梯度隐私. 该方法提出一种洪泛共识协议, 每轮随机选取某一参与方作为中央服务器完成全局聚合, 高效实现联邦学习的去中心化. 同时, 该方法引入 BLS 聚合签名, 防范外部敌手攻击, 提升验证效率. 理论分析和实验结果表明, 该方法是安全高效的, 相比同类联邦学习方法具有更高的效率.

关键词: 联邦学习; Shamir 秘密共享; 共识协议; BLS 聚合签名; 去中心化

引用格式: 邹洁丽,张子华,高铁杠.考虑外部敌手的去中心化联邦学习梯度聚合协议.计算机系统应用,2025,34(3):14–26. http://www.c-s-a.org.cn/1003-3254/9792.html

Decentralized Federated Learning Gradient Aggregation Protocol Considering External Adversaries

ZOU Jie-Li, ZHANG Zi-Hua, GAO Tie-Gang

(College of Software, Nankai University, Tianjin 300350, China)

Abstract: Federated learning is a distributed machine learning technique that allows participants to train models locally and upload updates to a central server. The central server aggregates the updates to generate a better global model, ensuring data privacy and solving the problem of data silos. However, the gradient aggregation relies on a central server, which may lead to a single point of failure, and the central server is also a potential malicious attacker. Therefore, federated learning needs to be decentralized. The existing decentralized solutions ignore external adversaries and the performance bottlenecks issues caused by data communication. To address the above issues, this study proposes a decentralized federated learning method considering external adversaries. The method applies Shamir's secret sharing scheme to divide model updates into multiple shares to protect gradient privacy. The method proposes a flooding consensus protocol that randomly selects a participant as the central server in each round to complete global aggregation, efficiently achieving the decentralization of federated learning. At the same time, the method introduces BLS aggregate signatures to prevent external adversary attacks and improve verification efficiency. Theoretical analysis and experimental results indicate that this method is safe and efficient, having higher efficiency than similar federated learning methods.

Key words: federated learning; Shamir's secret sharing; consensus protocol; BLS aggregate signature; decentralization

① 基金项目: 天津市自然科学基金重点项目 (21JCZDJC00130)
收稿时间: 2024-08-20; 修改时间: 2024-09-19; 采用时间: 2024-10-14; csa 在线出版时间: 2025-01-21
CNKI 网络首发时间: 2025-01-22

在当今数字化时代, 随着数据量的不断增长和数据来源的多样化, 机器学习由于获得更丰富的训练数据而不断进步. 然而, 当前数据分散在不同机构中. 由于数据隐私、数据标准和法律法规等原因, 大部分部门不愿意共享数据, 从而形成一座座数据孤岛, 阻碍机器学习的发展. 联邦学习作为一种分布式机器学习技术[1], 允许各个参与方在本地使用数据训练模型, 并将模型更新上传到中央服务器, 而不用共享原始数据, 从而保障各参与方的数据隐私和安全. 中央服务器聚合每轮的各个模型更新, 生成性能更好的全局模型, 解决了数据孤岛问题. 联邦学习已经广泛应用于智能物流[2–4]、金融服务[5–8]、交通管理[9–12]、医疗保健[13–16]和工业控制[17–20]等领域.

梯度聚合是联邦学习的核心步骤, 面临诸多挑战.

(1) 联邦学习依赖中央服务器完成梯度聚合. 这面临着单点故障问题, 也就是说一旦中央服务器出现故障, 整个联邦学习系统都将瘫痪. 并且, 由于中央服务器也是潜在的恶意攻击者, 所以存在不可信任问题. 尤其是在智能电网中[21], 分布式能源资源地理分散且通过稀疏通信链路连接; 在车载网络中[22], 车辆的通信范围有限且移动性高, 难以建立中央服务器. 因此, 需要设计一个适用于通信拓扑场景的联邦学习梯度聚合方案.

(2) 基于此, 许多可验证的中央服务器方案[23,24]和去中心化方案[25,26]被提出. 然而, 现有方案大多没有考虑到外部敌手攻击梯度问题. 参与方上传的模型更新暴露在网络中, 容易被窃取或者篡改. 攻击者若窃取到完整的梯度, 甚至可以通过成员推理攻击[27]和模型逆向攻击[28], 获知数据隐私. 攻击者若篡改模型更新, 将导致梯度聚合过程失效.

(3) 此外, 现有的去中心化验证方案存在大量的数据通信[26,29–31], 会导致联邦学习梯度聚合过程的性能瓶颈.

针对上述问题, 本文提出一种考虑外部敌手的去中心化联邦学习梯度聚合协议. 主要贡献如下.

(1) 在生成模型更新后, 应用 Shamir 秘密共享方案, 由梯度生成份额, 让份额在参与方之间通信, 保护模型更新的隐私, 即外部敌手无法获知完整的模型梯度且每个参与方只能获得部分参与方的模型梯度.

(2) 提出一种洪泛共识协议, 快速传播梯度份额, 并在每轮迭代中选出一个参与方代替中央服务器实现全局模型生成功能, 高效实现联邦学习的去中心化.

(3) 引入 BLS (Boneh-Lynn-Shacham) 聚合签名, 保证梯度份额的真实性, 防范外部敌手攻击, 并提升签名验证的效率.

1 相关工作

在联邦学习领域, 中心化和安全问题日益凸显, 许多研究工作致力于提出有效的解决方案.

针对中央服务器的现有工作[23,24,26], 大致可以分为两类. 一类是验证中央服务器的正确性. 文献[23]提出 zkFL, 一种基于零知识证明的联邦学习梯度聚合方法. 该方法利用零知识证明来确保聚合过程的正确性, 防止恶意的中央聚合器的操控. 该方法通过让聚合者在每轮提供证明, 增强系统的安全性, 但无法解决单点故障问题. 另一类是去中心化. 文献[24]提出一种基于区块链的去中心化联邦学习框架, 通过消除单点故障风险来增强系统的鲁棒性. 该框架利用区块链的透明性和不可篡改性, 确保参与者之间的信任. 然而, 区块链的引入将导致较高的计算和通信开销, 影响系统的整体效率. 结合 Shamir 秘密共享实现去中心化是可行的. 文献[26]提出一种在动态通信图中进行隐私保护的去中心化联邦学习方法. 该方法通过采用共识机制和秘密共享技术来实现安全的模型聚合, 能够有效应对高流动性环境下的通信挑战, 但它并未考虑外部敌手存在的情况.

现有的防御外部敌手攻击的工作, 有安全多方计算[29,30]、差分隐私[30]和数字签名[31]这 3 种. 文献[29]关注结合加密技术和安全多方计算的方案, 保护参与者的私有数据. 尽管该方法在数据保护方面表现出色, 但其实现复杂性和对计算资源的需求限制其在资源受限环境中的应用. 文献[30]则基于差分隐私和安全多方计算配置了一种名为 Privacyfl 的可扩展的模拟器, 旨在支持隐私保护和安全的联邦学习环境, 但其压缩和扰动会影响计算能力较低的边缘设备的效率. 数字签名是一种更加通用的解决方案. 文献[31]提出了一种基于群签名的高效隐私保护协议. 然而在现有的去中心化系统中, 广泛的数字签名会让系统的性能下降. 聚合签名验证是一种高效的方法, 可以显著减少验证所需的时间和存储空间. 因此, 在去中心化系统中, 聚合签名和 Shamir 秘密共享的结合是有价值的, 但是关于它们的研究尚缺.

2 背景知识

2.1 Shamir 秘密共享

Shamir 秘密共享是最早由 Shamir 在 1979 年基于拉格朗日多项式插值方法提出的算法[32], 具有可靠性和保护隐私等优点. 它的基本思想是分发者将一个秘密 s 分成 n 份子秘密, 并由 n 个参与者保管. 只有不少于 t 个参与者共同合作才能还原出秘密 s, 少于 t 个子秘密则无法获得关于秘密 s 的信息. 该技术包含以下算法.

(1) 份额生成算法 $GEN\_SHARES(s,n,t,p)=\{s_1,s_2,\cdots,s_n\}$: 输入秘密 s, 生成的份额数量 n, 还原秘密的最小份额数量 t, 大素数 p. 输出生成的份额集合 $\{s_1,s_2,\cdots,s_n\}$. 该算法通过构建一个如式 (1) 所示的 $t-1$ 次多项式 $f(x)$, 生成秘密份额 $s_i=f(i)$.

$$f(x)=a_{t-1}x^{t-1}+a_{t-2}x^{t-2}+\cdots+a_1x+s \bmod p \tag{1}$$

(2) 秘密还原算法 $RE\_SECRET(\{s_1,s_2,\cdots,s_t\},p)=s$: 输入任意一个不少于 t 个份额的集合 $\{s_1,s_2,\cdots,s_t\}$, 大素数 p. 输出还原的秘密 s. 该算法通过式 (2) 和式 (3) 还原秘密.

$$f(x)=\sum_{i=1}^{t}\left(s_i\times\prod_{1\leqslant j\leqslant t,\,i\neq j}\frac{x-j}{i-j}\right) \tag{2}$$

$$s=f(0)=\sum_{i=1}^{t}\left(s_i\times\prod_{1\leqslant j\leqslant t,\,i\neq j}\frac{-j}{i-j}\right) \tag{3}$$

2.2 BLS 聚合签名

BLS 聚合签名是最早由 Boneh 等基于双线性映射提出的签名算法[33]. 它可以将多个签名聚合成一个签名进行验证, 减少通信开销和加快签名验证, 具有签名短和安全性高等优点. 该技术包含以下算法.

(1) 初始化: 双线性映射 $e:G_1\times G_2\to G_T$, 哈希函数 $H:\{0,1\}^*\to G_2$.

(2) 密钥生成算法: 随机选择私钥 sk, 计算公钥 $pk=g^{sk}$.

(3) 签名算法 $SIGN(sk,m)=sig$: 输入私钥 sk 和消息 m. 输出签名 sig. 计算 $h=H(m)$ 和 $sig=h^{sk}$.

(4) 聚合签名算法 $AGG(\{sig_1,sig_2,\cdots,sig_k\})=agg$: 输入多个签名 $\{sig_1,sig_2,\cdots,sig_k\}$. 输出聚合签名 agg.

(5) 验证算法 $VERIFY(\{pk_1,pk_2,\cdots,pk_k\},\{m_1,m_2,\cdots,m_k\},agg)=result$. 输入多个公钥 $\{pk_1,pk_2,\cdots,pk_k\}$, 多个消息 $\{sig_1,sig_2,\cdots,sig_k\}$ 和聚合签名 agg. 输出验证结果 $result$. 若等式 (4) 成立, 则验证成功; 反之, 验证失败.

$$\begin{aligned} &e(sig,g)=\sum_{i=1}^{k}e(h_i,pk_i)\\ &\Downarrow\\ &e\left(\sum_{i=1}^{k}h_i^{sk_i},g\right)=\sum_{i=1}^{k}e\left(h_i,g^{sk_i}\right)\end{aligned} \tag{4}$$

3 方案描述

本节中详细介绍了提出的系统. 首先描述该系统的总体框架, 然后介绍 Shamir 秘密共享方案, 接着介绍实现去中心化的洪泛共识协议, 最后引入 BLS 聚合签名算法防范外部敌手攻击.

3.1 总体框架

本文提出的系统由多个客户端组成, 每个客户端拥有自己的本地数据集, 共同训练同一个机器学习模型. 该系统的工作流程如图 1 所示.

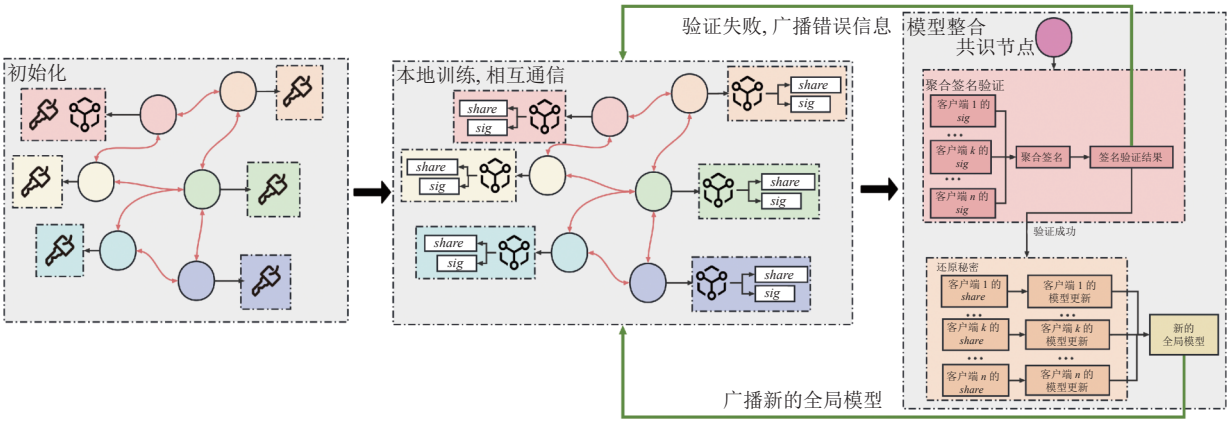


图 1 系统的工作流程

(1) 初始化: 某一客户端随机初始化全局模型, 并广播全局模型. 同时, 所有客户端广播自己的公钥. 其中, 公钥用于验证份额签名.

(2) 本地训练: 各客户端获得全局模型后, 使用本地数据集训练本地模型, 得到模型梯度. 然后, 各客户端生成模型梯度对应的份额集合和份额签名集合. 份额生成和签名方法将在第 3.2 和 3.4 节中详细论述.

(3) 相互通信: 根据洪泛共识协议, 各客户端分发自己的份额和份额签名, 传播更新的相应集合, 直到选出共识节点. 洪泛共识协议将在第 3.3 节中详细论述.

(4) 模型聚合: 共识节点首先聚合各客户端的份额签名集合, 生成聚合签名, 并验证聚合签名. 验证通过后, 共识节点还原各客户端的模型梯度, 实现全局聚合, 生成新的全局模型. 模型梯度还原和聚合签名方法将在第 3.2 和 3.4 节中详细论述.

(5) 重复迭代: 共识节点广播全局模型后, 重复迭代步骤 (2)–步骤 (4), 直到全局模型到达理想性能或者迭代次数到达规定次数.

3.2 Shamir 秘密共享方案

联邦学习中, 客户端传输的本地梯度暴露在网络中, 容易被系统外敌人攻击. 为了让梯度信息在分布式网络环境中安全共享, 所提出的系统采用 Shamir 秘密共享方案, 将本地梯度当作秘密, 拆分成多个份额, 并传输份额给其他客户端. 外部敌手即使非法获取了某一份额, 也无法还原秘密, 得到梯度信息. 考虑到联邦学习中传输的梯度是形如<变量名, 张量值>的字典, 该系统将 Shamir 秘密共享方案变形, 以便更好地应用于联邦学习中, 实现系统所需功能.

设计的份额生成算法如算法 1 所示.

算法 1. Shamir 秘密共享方案的份额生成算法

输入: 梯度信息 *secret_gradients*; 生成份额数量 *num_shares*; 还原秘密的最小份额数量 *threshold*; 大素数 *prime*.
输出: 生成的份额集合 *shares*.

1) for *key*, *secret_gradient* in *secret_gradients*:
    share[*key*]←*GEN_SHARES*(*secret_gradient*, *num_shares*, *threshold*, *prime*);
2) 输出生成的份额集合 *shares*.

设计的秘密还原算法如算法 2 所示.

算法 2. Shamir 秘密共享方案的秘密还原算法

输入: 大素数 *prime*; 值数量达到 *threshold* 的份额集合 *subset_of_shares*.
输出: 还原的梯度信息 *secret_gradients*.

1) for *key*, *share* in *subset_of_shares*:
    secret_gradients[*key*]←*RE_SECRET*(*share*, *prime*);
2) 输出还原的梯度信息 *secret_gradients*.

如图 2 所示, 联邦学习中, 每个客户端使用本地数据集来训练本地模型, 得到本地梯度, 然后使用算法 1 得到本地梯度中每个变量名对应的张量值的份额集合.

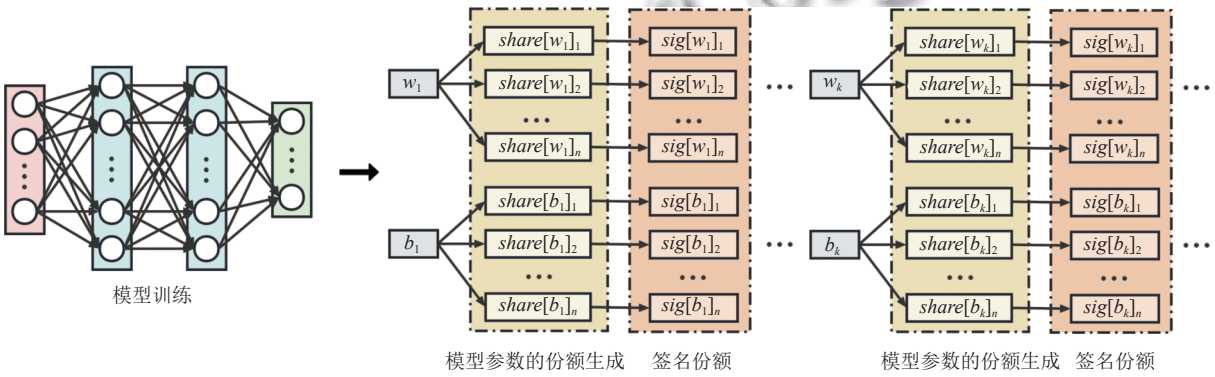


图 2 份额生成和签名

3.3 洪泛共识协议

联邦学习依赖中央服务器聚合所有客户端的梯度, 这存在单点故障问题. 由于客户端和客户端之间可以相互通信, 形成网络拓扑图, 所以采取一种共识协议选出某一客户端代替中央服务器完成聚合功能. 本文提出一种名为洪泛共识协议的共识算法, 结合 Shamir 秘密共享方案, 随机选出客户端实现全局聚合. 这种客户端选择的随机性, 有效消除了单点故障. 洪泛共识协议如算法 3 所示.

算法 3. 洪泛共识协议

1) 客户端集合 *clients* 使用算法 1, 生成本地梯度的份额集合, 并根据份额集合 *client.shares* 的数量和邻居客户端 *client.neighbors* 的数量, 向邻居发送一定份的份额. 具体算法如下:

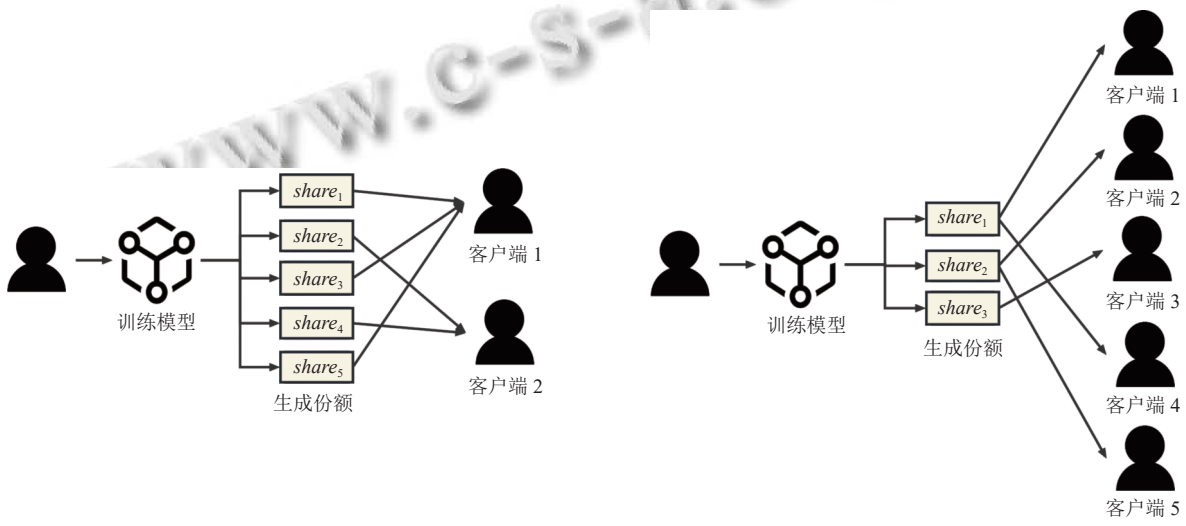
```
for client in clients:
    client.shares←client.算法 1();
    n←0;
    if Num(client.shares)<Num(client.neighbors):
        for neighbor in client.neighbors:
            client.send(neighbor, client.shares[n++]);
            if n>Num(client.shares):
                n←0;
    else:
        for share in client.shares:
            client.send(client.neighbors[n++], share);
            if n>Num(client.neighbors):
                n←0;
```

2) 当客户端 *client* 接收到邻居客户端 *neighbor* 发送的份额 *share* 后, 更新自己用于存储其他客户端份额的集合 *nbs_shrs*, 然后将更新分享给自己的邻居们, 加快共识速度. 选出的共识节点, 使用算法 2 还原梯度信息, 实现全局聚合. 具体算法如下:

```
client.nbs_shrs[neighbor].add(share);
if Num(client.nbs_shrs[neighbor]) == threshold:
    client.nbs_grads.add(neighbor);
if Num(client.nbs_grads) == Num(clients):
    for neighbor in client.neighbors:
        client.send(neighbor, “Stop sending”);
        client.nbs_grads[neighbor]←client.算法 2();
    global_grad←client.aggregate(client.nbs_grads);
    client.send(client.neighbors, global_gradient);
else:
    for neighbor in client.neighbors:
        client.send(neighbor, share);
```

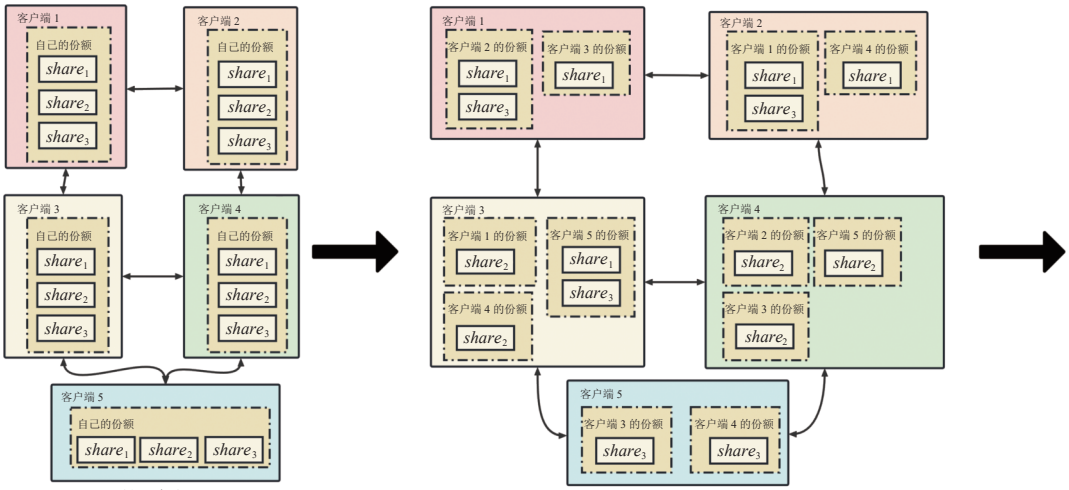
图 3 和图 4 展示了该系统结合洪泛共识协议和 Shamir 秘密共享方案实现去中心化的工作流程. 图 3 具体描述了客户端向邻居分发份额的方法. 当客户端数量小于份额数量时, 以邻居数为 2 和份额数为 5 举例, 每个份额依次向邻居发放, 一个客户端可以获得多个份额, 直到份额全部发放完毕. 当客户端数量大于份额数量时, 以邻居数为 5 和份额数为 3 举例, 每个份额依次向邻居发放, 每个份额可以重复发放给客户端, 直到所有邻居都获得份额. 图 3 展示了份额分发的一种情况, 实际上份额和客户端的对应是随机的.

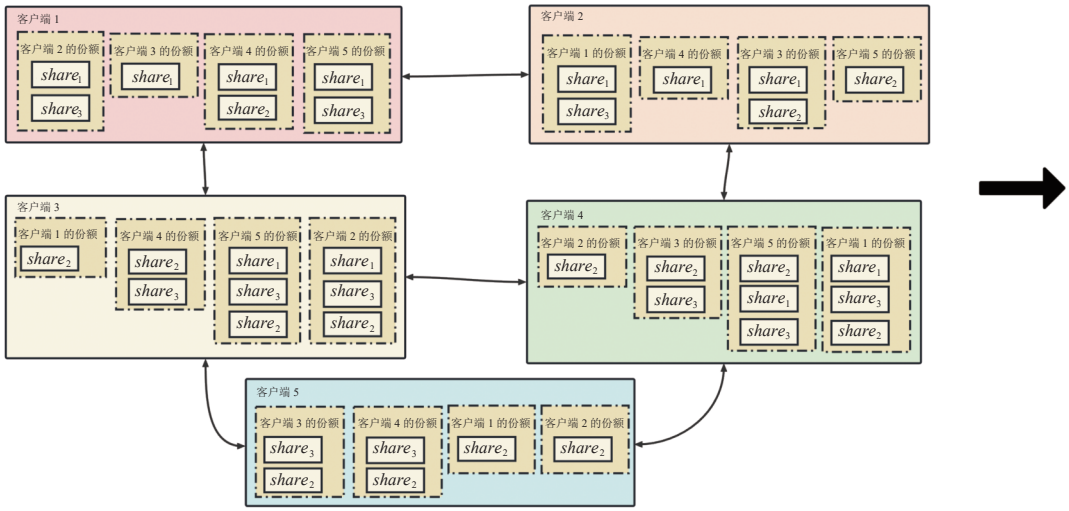
图 4 具体描述了从分发份额到选出共识节点的一种情况. 以客户端数为 5 和份额数为 3 为例, 假设还原秘密的最小份额数等于 3, 各客户端通信距离和通信速度等完全一致. 第 1 阶段, 各客户端生成自己的 3 个份额; 第 2 阶段, 各客户端根据份额分发规则向邻居分发自己的份额, 此时各客户端获得邻居的部分份额; 第 3 阶段, 各客户端由于存储其他客户端份额的集合更新了, 向邻居分享更新, 此时各客户端获得邻居的部分份额; 第 4 阶段, 某一客户端获得的其他所有客户端的份额数均不小于 3, 该客户端被选作共识节点. 在复杂的网络通信拓扑图中, 第 3 阶段将进行多轮通信, 因为一旦某一客户端存储其他客户端份额的集合更新, 就将进行一次与邻居客户端的通信. 在第 4 阶段, 客户端 3 和客户端 4 都将能被选作共识节点. 但在实际过程中, 各客户端情况并不完全一致, 某一客户端将率先被选作共识节点.



(a) 当客户端数量小于份额数量, 份额分发方法　　(b) 当客户端数量大于份额数量, 份额分发方法

图 3　份额分发方法





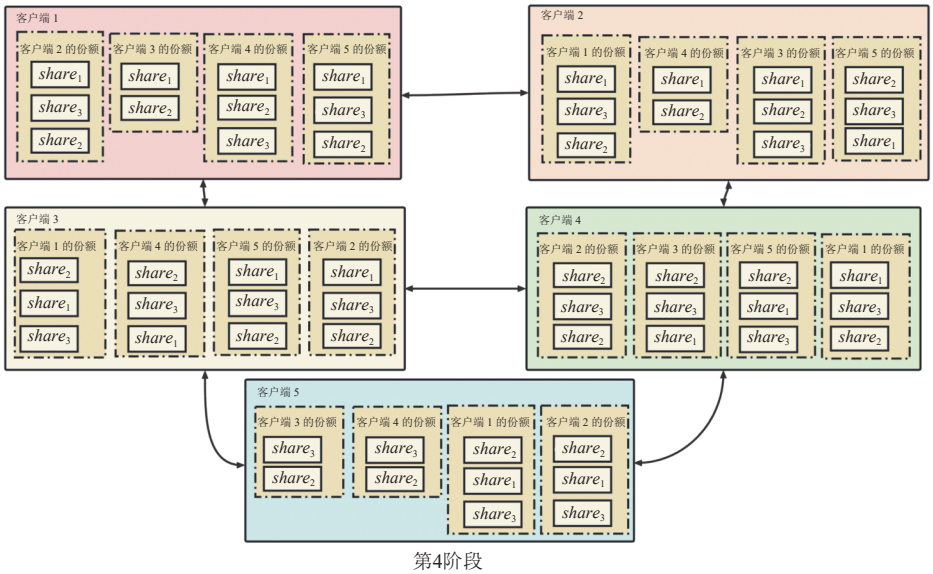


图 4 共识节点的选出过程

3.4 BLS 聚合签名

考虑外部敌手, 所提出的系统引入数字签名技术, 验证发送信息的真实性. 由于上述的秘密共享和共识将消耗大量的通信带宽, 所以采用 BLS 聚合签名方法减少签名验证的通信代价. 签名验证工作由共识节点完成, 而其余节点无需验证签名, 这样可以提高工作效率. 针对本地梯度份额的 BLS 聚合签名方法如算法 4 所示.

算法 4. BLS 聚合签名方法

1) 每个客户端使用自己的私钥 *sk* 签名自己的份额集合 *shares*, 获得签名份额集合 *sigs*, 即:
for *key*, *values* in *shares*:
  for *value* in *values*:
    $sigs[key]_i \leftarrow SIGN(sk, value)$;
2) 通过洪泛共识协议, 选出共识节点;
3) 共识节点聚合自己获得的其他所有客户端的份额签名集合 *list_sigs*, 获得聚合签名 $agg \leftarrow AGG(list\_sigs)$;
4) 共识节点使用各个客户端的公钥集合 *list_pk* 和拥有的其他所有客户端的份额集合 *list_shares*, 验证聚合签名, 获得验证结果 $result \leftarrow VERIFY(list\_pk, list\_shares, agg)$;
5) 若 *result* 为正确, 共识节点还原梯度信息, 并实现全局聚合; 反之, 向其他客户端广播错误信息, 要求重传.

算法 4 可以高效验证份额的正确性, 防范外部敌手攻击, 份额签名流程如图 2 所示. 当模型参数的份额生成后, 签名对应的份额.

4 理论分析

4.1 安全性分析

4.1.1 隐私性分析

定义 1 (多项式插值难题). 给定一组点 (x_1,y_1), $(x_2,y_2),\cdots,(x_k,y_k)$, 其中 x_i 是不同的实数, y_i 是对应的函数值. 在 k 个点的情况下, 找到多项式 $f(x)$ 的系数是可行的, 但是在少于 k 个点的情况下, 在多项式时间内无法唯一确定 $f(x)$.

断言 1. 在所提方案的梯度聚合过程中, 原始的梯度信息是被保护的.

证明: 在提出的系统中, 客户端向邻居传输梯度份额 $\{s_1,s_2,\cdots,s_n\}$ 中的部分, 其中还原阈值为 k, 直到选出共识节点. 显而易见的是, 梯度的直接信息并未暴露在网络中且没有直接共享给任何参与者. 假设任意两个客户端之间的网络传输过程存在外部敌手, 并且敌手获得传输信息 s_t. 根据洪泛共识协议, 敌手一次仅能获得其中一份. 若敌手成功推知原始的梯度信息, 那么将与多项式插值难题相矛盾. 此外, 在选出共识节点前, 各客户端至少有一个其他客户端的梯度份额数少于 k. 若除共识节点外的参与方获知所有的原始梯度信息, 那么将与多项式插值难题相矛盾. 考虑到实际情况, 各参与方获知的其他梯度信息将更少. 因此, 在所提方案的梯度聚合过程中, 原始的梯度信息是被保护的.

4.1.2 抗外部敌手攻击分析

定义 2 (离散对数问题 (discrete logarithm problem, DLP)[34]). 给定一个有限的循环群 G, 以及群的生成元 g 和群元素 g^x, 在多项式时间内计算出 x 是不可行的.

定义 3 (Diffie-Hellman 计算 (computational Diffie-Hellman, CDH)[35]). 给定群 G, 其生成元为 g. 已知 g^a, g^b, 在多项式时间内计算出 g^{ab} 是不可行的.

断言 2. 所提方案可以抵御外部敌手攻击.

证明: 假设存在外部敌手, 成功伪造一个 BLS 签名, 那么敌手必然获知某一私钥 *sk*. 但是在已知公钥 *pk* 和签名 $H(m)$ 的情况下, 敌手成功破解 *sk* 将与 DLP 相矛盾. 若敌手能够伪造聚合签名, 那么敌手将构造出有效的等式 $e(sig,g)=e(H(m),pk)$, 也就是在已知 g^a, g^b 的情况下成功计算出 g^{ab}, 这与 DLP 相矛盾. 此外, 在提出的系统中, 每个客户端都会获知其他部分客户端的签名. 敌手若攻击成功, 那么必须在有限的网络传输时间内同时成功向所有客户端发送伪造签名. 在实际场景中, 这不可能发生. 任一拥有公钥的参与方都能识别出签名是否伪造. 而且每轮的共识节点是随机的, 与通信距离和带宽等有关, 敌手无法预测来影响全局聚合的结果.

4.2 时间复杂度分析

假设 Shamir 秘密共享方案生成份额的时间为 *sg*, 还原秘密的时间为 *sr*. BLS 聚合签名生成签名的时间为 *bg*, 聚合签名的时间为 *ba*, 验证聚合签名的时间为 *bv*. 模型梯度共有 *nw* 个参数, 生成的份额数为 *ns*. 本文方案的算法时间消耗如表 1 所示.

表 1 本文方案的算法时间消耗

本文方案的算法	时间
生成份额	$nw\times sg$
还原份额	$nw\times sr$
签名份额	$nw\times ns\times bg$
验证份额	$ba+bv$

分析洪泛共识协议的时间复杂度, 假设客户端数量为 n, 每个客户端的邻居节点的数量为 $\{m_1,m_2,\cdots,m_n\}$. 对于第 k 个客户端, 分发份额给邻居客户端的时间复杂度为 $O(\max(m_k,ns))$. 分发份额后, 当收到新的份额, 第 k 个客户端向邻居发送份额, 时间复杂度为 $O(m_k)$. 若有 t_k 次更新, 则时间复杂度为 $O(t_k\times m_k)$. 第 k 个客户

端从分发份额, 到传播新的份额, 再到结束的时间复杂度为$O(\max(m_k,ns)+t_k\times m_k)$. 由于 n 个客户端是并行传输的, 所以总体时间复杂度为如式 (5) 所示.

$$O\left(\max_{1\leqslant k\leqslant n}(\max(m_k,ns)+t_k\times m_k)\right) \tag{5}$$

5 实验分析

5.1 实验设置

本研究实验环境为 Ubuntu 20.04, Intel(R) Xeon(R) Platinum 8358P CPU@2.60 GHz 处理器, 24 GB RTX 3090 显卡, 80 GB 内存, 80 GB 硬盘, 编程语言为 Python, 实验框架基于 PyTorch. 本文实验基于两个通用的机器学习任务图像分类与语言理解展开.

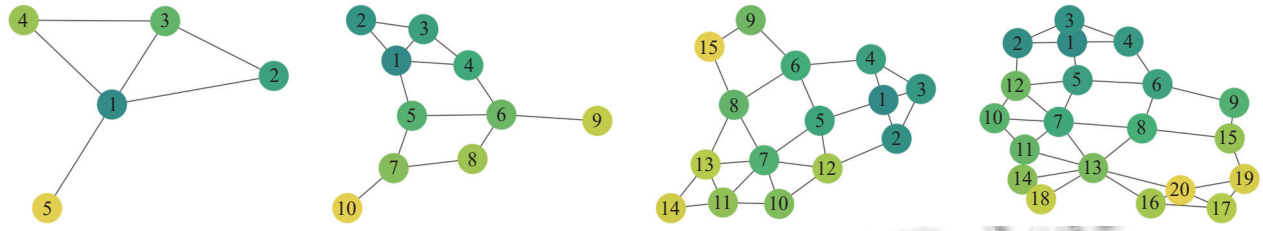
图像分类是一项基准计算机视觉任务. 本文采用 CIFAR-10 数据集. 该数据集共有 50 000 张训练图片和 10 000 张测试图片, 包含 10 个类别的 32×32 彩色图片 (飞机、汽车、鸟类、猫、鹿、狗、蛙类、马、船和卡车). 本文使用 ResNets (ResNet18, ResNet34 和 ResNet50)[36]和 DenseNets (DenseNet121, DenseNet169 和 DenseNet201)[37]模型进行测试.

语言理解是一项基准自然语言处理任务. 本文采用 PTB (penn treebank) 数据集. 该数据集共有 3 个 txt 文件, 分别作为训练集 (train)、验证集 (valid) 和测试集 (test). 这 3 个文件分别包含 42 000、3 000 和 3 000 句英文. 本文使用单层、双层和三层 LSTM (long short-term memory)[38]模型进行测试.

5.2 实验结果分析

5.2.1 正确性分析

为了证明本文方案可以正确实现梯度聚合, 本文设置 5, 10, 15 和 20 个客户端, 并假设它们相互通信的拓扑图如图 5 所示. 在不同的机器学习任务场景下, CIFAR-10 与 PTB 数据集随机分配给各个客户端. 每个客户端使用本地数据集单独训练各自的模型, 然后将本地梯度作为秘密分成 3, 6, 9, 12 或 15 个份额, 依次发送给自己的邻居客户端. 其中, 还原秘密的最少份额数量为生成的份额数减 2. 表 2 列出了有关正确性的部分实验结果, 表明在不同参数的多次实验中梯度聚合均能成功实现.



(a) 5 个客户端 (b) 10 个客户端 (c) 15 个客户端 (d) 20 个客户端

图 5 客户端的通信拓扑图

表 2 正确性情况

客户端数量	份额数量	模型	正确聚合梯度	客户端数量	份额数量	模型	正确聚合梯度	客户端数量	份额数量	模型	正确聚合梯度
10	9	ResNet18	是	10	9	LSTM1	是	20	9	DenseNet121	是
10	15	ResNet18	是	10	15	LSTM1	是	20	15	DenseNet121	是
10	9	ResNet50	是	10	9	LSTM3	是	20	9	DenseNet201	是
10	15	ResNet50	是	10	15	LSTM3	是	20	15	DenseNet201	是
10	9	DenseNet121	是	20	9	ResNet18	是	20	9	LSTM1	是
10	15	DenseNet121	是	20	15	ResNet18	是	20	15	LSTM1	是
10	9	DenseNet201	是	20	9	ResNet50	是	20	9	LSTM3	是
10	15	DenseNet201	是	20	15	ResNet50	是	20	15	LSTM3	是

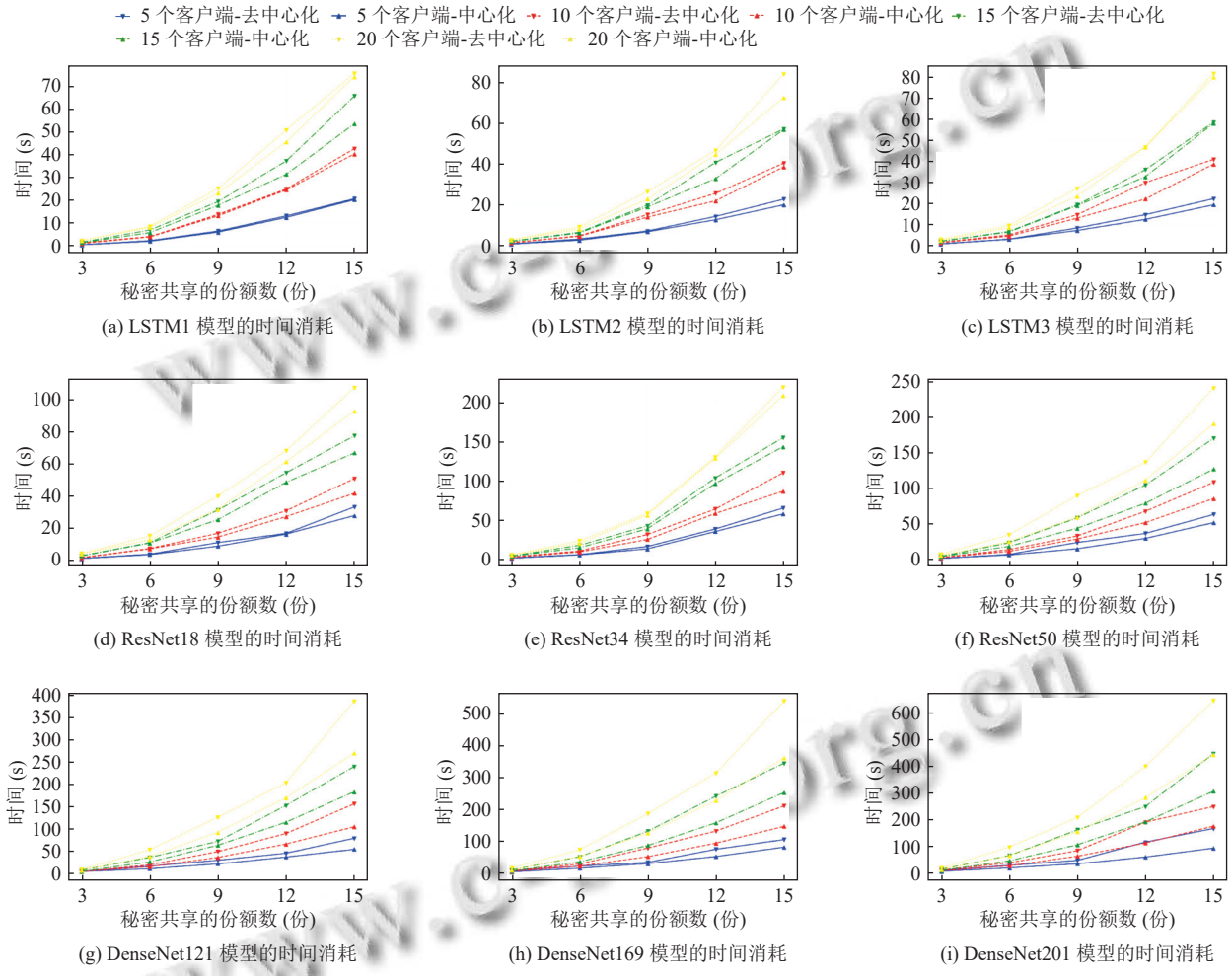
5.2.2 性能分析

本节完成以下实验内容: (1) 比较中心化和去中心化场景下, 客户端开始通信到某一客户端开始全局聚合的时间消耗; (2) 不同客户端数量下, 客户端开始通信到某一客户端开始全局聚合的通信轮数; (3) 本文方案与其他方案的时间对比. 通过上述性能分析, 可以发现本文方案与其他中心化的方案相比, 具有相似的效率. 本文方案的通信轮数是近似线性增加的, 并且与其他同类去中心化方案相比, 具有更优的时间效率.

5.2.2.1 中心化和去中心化联邦学习的时间消耗

图 6 展示了通信不同模型时, 不同数量的客户端随着秘密共享的份额数增加的时间消耗的情况. 随着

客户端数量和份额数的增多, 时间消耗也逐渐增大. 当客户端数量低于 15 时, 中心化和去中心化的时间消耗差距不大; 当客户端数量高于 15 时, 中心化和去中心化的时间消耗将有显著差距. 比较图 6(a)–图 6(e) 和图 6(f)–图 6(i) 可知, LSTM、ResNet18 和 ResNet34 模型的时间消耗差距较小, 而 ResNet50 和 DenseNets 模型的去中心化时间消耗将近似高于中心化时间消耗的 1/3. 这是由于 ResNet50 和 DenseNets 模型的模型参数达到 300 个以上, 并且 Shamir 秘密共享方案生成份额的时间是随着模型参数数量而线性增长的, 大数据的传输也较慢, 所以整体时间消耗增多. 但当份额数低于 9 时, 这种时间消耗差距并不显著, 所以对于较为简单的模型, 可以设置较高的份额数; 对于较为复杂的模型, 可以设置较低的份额数, 保证时间效率.



(a) LSTM1 模型的时间消耗 (b) LSTM2 模型的时间消耗 (c) LSTM3 模型的时间消耗

(d) ResNet18 模型的时间消耗 (e) ResNet34 模型的时间消耗 (f) ResNet50 模型的时间消耗

(g) DenseNet121 模型的时间消耗 (h) DenseNet169 模型的时间消耗 (i) DenseNet201 模型的时间消耗

图 6 不同场景和模型的时间消耗

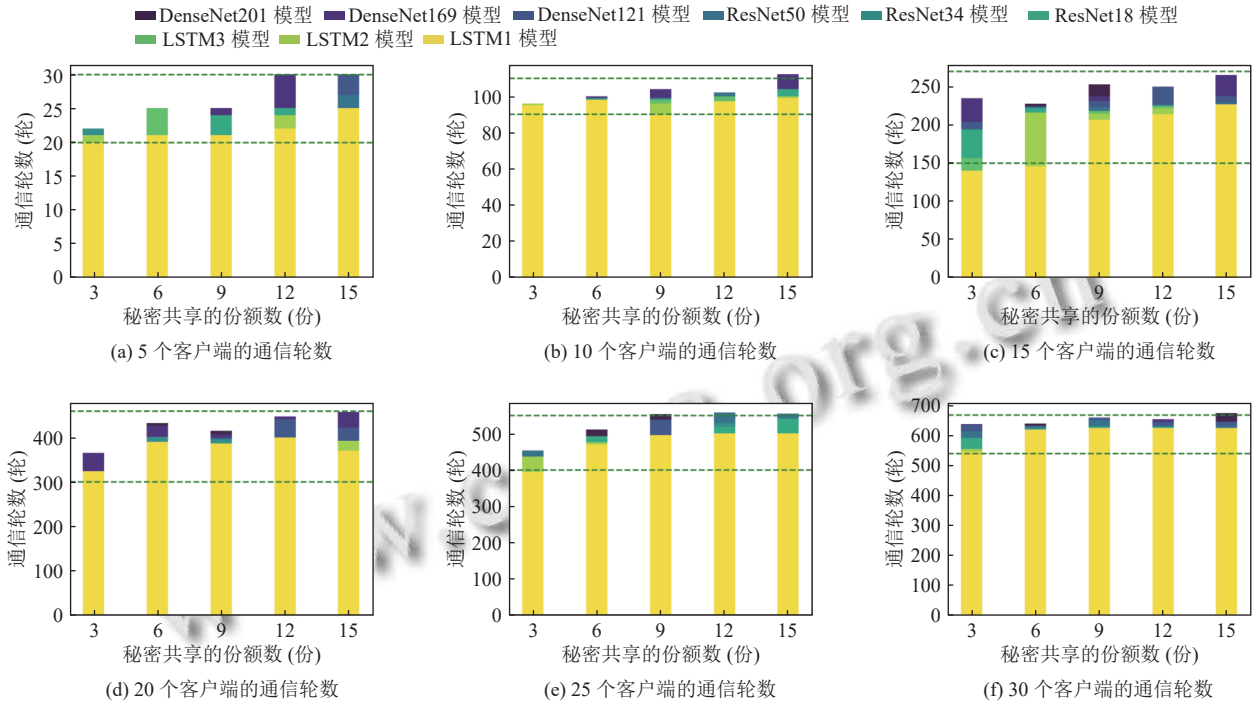
5.2.2.2 不同客户端数量下联邦学习的通信轮数

图 7 柱状图展示了不同数量的客户端, 通信不同模型时随着秘密共享的份额数增加的通信轮数的情况. 不同模型的通信轮数近似相同. 在去中心化的情况下, 5 个客户端的通信轮数为 20–30 轮; 10 个客户端的通信轮数为 90–110 轮; 15 个客户端的通信轮数为 150–250 轮; 20 个客户端的通信轮数为 300–440 轮; 25 个客户端的通信轮数为 400–550 轮; 30 个客户端的通信轮数为 540–670 轮. 可以推测的是, 每增加一个客户端, 通信轮数将增加 20 轮左右. 一般共识协议, 客户端之间需要两两通信, 导致通信轮数是指数增加的. 本文方案的通信轮数是近似线性增加的, 相比一般共识协议具有更好的性能.

5.2.2.3 与其他方案对比

图 8 展示了本文方案与文献[29]方案的时间对比. 由于文献[29]方案为全联通通信场景, 故对比实验在不同数量的客户端的全联通场景下, 并设置份额数量为 9 个. 本文方案的联邦学习效率高于文献[29]方案, 这

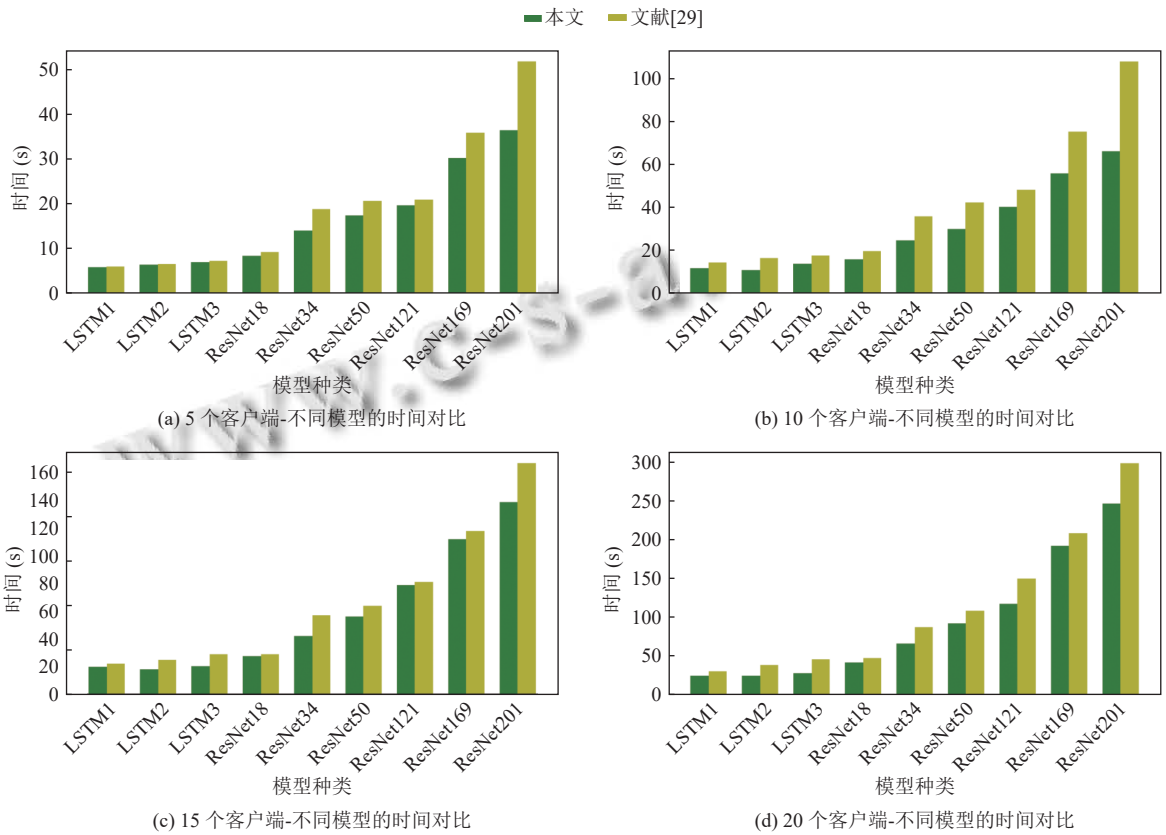
是由于本文采用洪泛共识协议, 当其中一个客户端获得足够多的份额后, 就作为共识节点完成全局聚合, 避免了多余的份额发放. 而且文献[29]需要考虑模型参数的批次划分, 增加了份额产生和分发的时间.



(a) 5 个客户端的通信轮数 (b) 10 个客户端的通信轮数 (c) 15 个客户端的通信轮数

(d) 20 个客户端的通信轮数 (e) 25 个客户端的通信轮数 (f) 30 个客户端的通信轮数

图 7 不同场景和模型的通信轮数



(a) 5 个客户端-不同模型的时间对比 (b) 10 个客户端-不同模型的时间对比

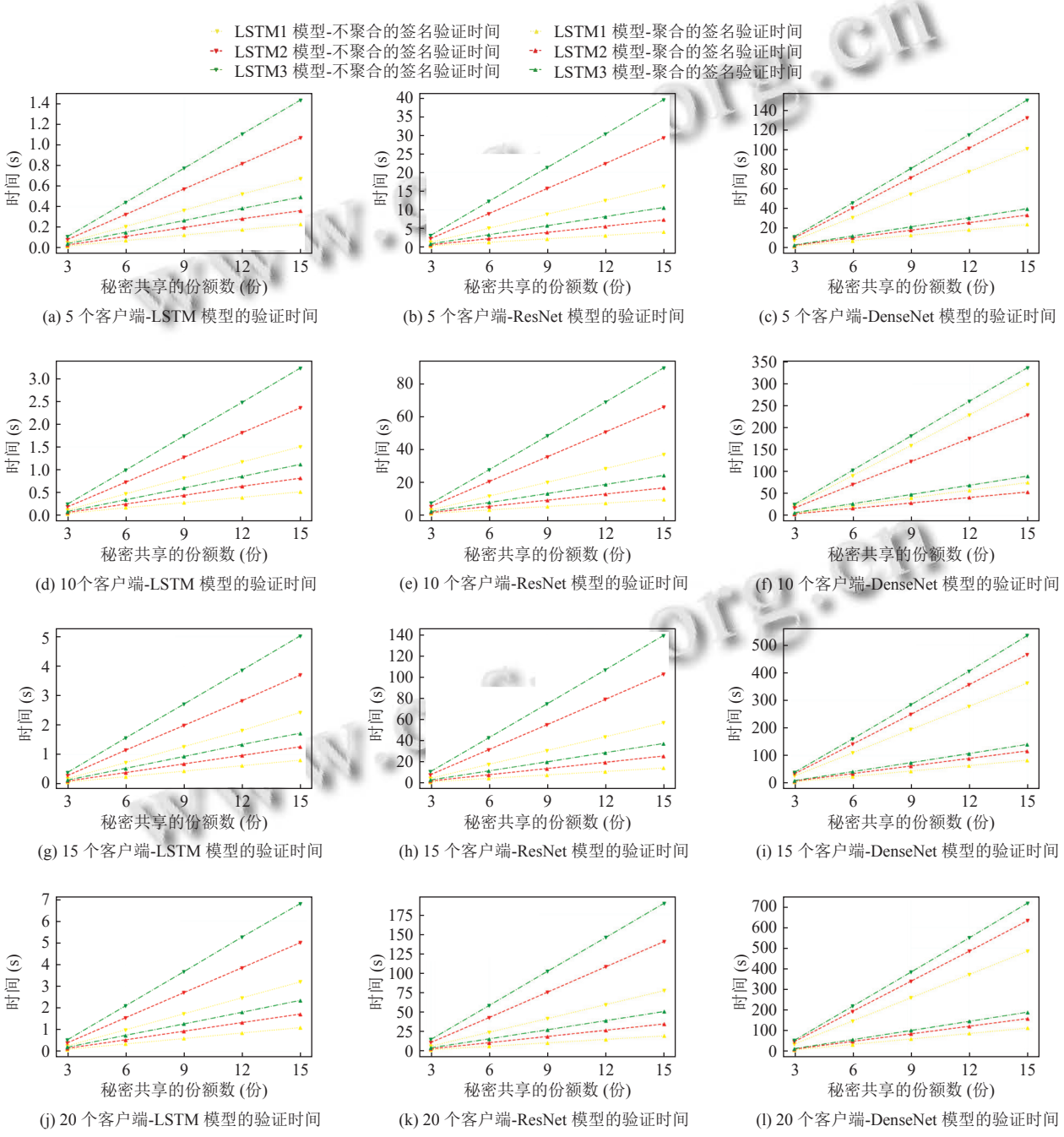
(c) 15 个客户端-不同模型的时间对比 (d) 20 个客户端-不同模型的时间对比

图 8 本文方案与其他方案的效率对比

5.2.3 安全性分析

由于洪泛共识协议消耗了较多时间, 所以采用 BLS 聚合签名的方法防御外部攻击. 每个客户端依次签名自己的每个份额, 并将份额签名随着对应的份额发送给邻居客户端. 当某一客户端获得足够多的其他所有客户端的份额数后, 聚合所有份额签名, 并验证所有份额的聚合签名是否正确.

图 9 展示了采用和不采用 BLS 聚合签名的情况. 采用 BLS 聚合签名的验证时间为签名聚合和聚合签名验证的时间之和. 不采用聚合签名的验证时间为每个份额单独签名验证的时间之和. 可以看出, 签名验证时间随着份额数的增多而线性增长, 且 BLS 聚合签名的验证时间远低于不采用聚合签名的验证时间. 这种时间效率的提升随着客户端数量的增多和模型复杂度的增加而越来越明显, 所以 BLS 聚合签名可以显著提高效率.



(a) 5 个客户端-LSTM 模型的验证时间 (b) 5 个客户端-ResNet 模型的验证时间 (c) 5 个客户端-DenseNet 模型的验证时间

(d) 10个客户端-LSTM 模型的验证时间 (e) 10 个客户端-ResNet 模型的验证时间 (f) 10 个客户端-DenseNet 模型的验证时间

(g) 15 个客户端-LSTM 模型的验证时间 (h) 15 个客户端-ResNet 模型的验证时间 (i) 15 个客户端-DenseNet 模型的验证时间

(j) 20 个客户端-LSTM 模型的验证时间 (k) 20 个客户端-ResNet 模型的验证时间 (l) 20 个客户端-DenseNet 模型的验证时间

图 9 签名验证时间对比

6 结论与展望

本文提出一种考虑外部敌手的去中心化联邦学习方法. 该方法提出洪泛共识协议, 并应用 Shamir 秘密共享方案, 实现联邦学习的去中心化. 同时, 考虑到去中心化方法消耗了大量时间, 该方法采用 BLS 聚合签名的方法防御外部攻击. 实验结果表明, 该方法是正确有效的. 在客户端数量较少时, 可以拥有与中心化联邦学习相似的效率. 在训练较为复杂的模型时, 该方法可以通过设置较少的份额数, 保证通信效率. 同时, 在客户端通信拓扑图较为稀疏的情况下, 该方法表现出良好的性能. 在防御外部攻击上, 该方法的效率提升随着客户端数量的增多和模型复杂度的增加而越发显著.

在后续的研究工作中, 将关注更复杂的客户端通信拓扑图, 设计出应用更广泛和性能更好的框架.

参考文献

1 McMahan HB, Moore E, Ramage D, *et al*. Communication-efficient learning of deep networks from decentralized data. Proceedings of the 20th International Conference on Artificial Intelligence and Statistics. Fort Lauderdale: PMLR, 2017. 1273–1282.

2 王攀, 缪祥华. 联邦学习在船舶物流安全领域的应用. 中国水运, 2024(6): 74–76. [doi: 10.13646/j.cnki.42-1395/u.2024.06.026]

3 Zhang C, Liu X, Xu J, *et al*. An edge based federated learning framework for person re-identification in UAV delivery service. Proceedings of the 2021 IEEE International Conference on Web Services (ICWS). Chicago: IEEE, 2021. 500–505. [doi: 10.1109/ICWS53863.2021.00070]

4 Supriya Y, Srivastava G, Dasaradharami Reddy K, *et al*. PSO-enabled federated learning for detecting ships in supply chain management. Proceedings of the 30th International Conference on Neural Information Processing. Changsha: Springer, 2023. 413–424. [doi: 10.1007/978-981-99-8132-8_31]

5 林宏峥, 金维国, 宋国英, 等. 基于金融场景数据流通的安全技术研究. 网络安全技术与应用, 2024(3): 105–107. [doi: 10.3969/j.issn.1009-6833.2024.03.039]

6 张卓. 基于联邦学习的数据隐私权保护研究——以微众银行、平安科技等为例的分析. 国外社会科学前沿, 2024(5): 86–99.

7 Long GD, Tan Y, Jiang J, *et al*. Federated learning for open banking. In: Yang Q, Fan LX, Yu H, eds. Federated Learning: Privacy and Incentive. Cham: Springer, 2020. 240–254. [doi: 10.1007/978-3-030-63076-8_17]

8 Byrd D, Polychroniadou A. Differentially private secure multi-party computation for federated learning in financial applications. Proceedings of the 1st ACM International Conference on AI in Finance. New York: ACM, 2020. 16. [doi: 10.1145/3383455.3422562]

9 李志, 林森, 张强. 面向轨道交通智能故障检测联邦学习模型的云边协同训练方法. 计算机科学, 1–12. http://kns.cnki.net/kcms/detail/50.1075.tp.20240625.1100.026.html. (2024-06-26)[2024-08-05].

10 乐俊青, 谭州勇, 张迪, 等. 面向车联网数据持续共享的安全高效联邦学习. 计算机研究与发展, 2024, 61(9): 2199–2212. [doi: 10.7544/issn1000-1239.202330894]

11 Manias DM, Shami A. Making a case for federated learning in the internet of vehicles and intelligent transportation systems. IEEE Network, 2021, 35(3): 88–94. [doi: 10.1109/MNET.011.2000552]

12 Lu YL, Huang XH, Zhang K, *et al*. Blockchain empowered asynchronous federated learning for secure data sharing in internet of vehicles. IEEE Transactions on Vehicular Technology, 2020, 69(4): 4298–4311. [doi: 10.1109/TVT.2020.2973651]

13 张连福, 谭作文. 一种面向多模态医疗数据的联邦学习隐私保护方法. 计算机科学, 2023, 50(S2): 230800021. [doi: 10.11896/jsjkx.230800021]

14 陆枫, 李炜, 顾琳, 等. 基于迭代协作学习框架的信誉医学参与方选择. 计算机研究与发展, 2024, 61(9): 2347–2363. [doi: 10.7544/issn1000-1239.202330270]

15 Antunes RS, da Costa CA, Küderle A, *et al*. Federated learning for healthcare: Systematic review and architecture proposal. ACM Transactions on Intelligent Systems and Technology (TIST), 2022, 13(4): 54. [doi: 10.1145/3501813]

16 Li JC, Meng Y, Ma LC, *et al*. A federated learning based privacy-preserving smart healthcare system. IEEE Transactions on Industrial Informatics, 2022, 18(3): 2021–2031. [doi: 10.1109/TII.2021.3098010]

17 吴维鑫, 侯会文, 石乐义. 基于深度学习和联邦学习的工控入侵检测研究. 微电子学与计算机, 2024, 41(9): 22–31.

18 李健俊, 王万江, 陈鹏, 等. 基于联邦学习的工控机业务行为分布式安全检测. 计算机集成制造系统, 1–23. http://kns.cnki.net/kcms/detail/11.5946.TP.20231023.0904.004.html. (2023-10-13)[2024-08-05].

19 Huong TT, Bac TP, Long DM, *et al*. Detecting cyberattacks using anomaly detection in industrial control systems: A federated learning approach. Computers in Industry, 2021, 132: 103509. [doi: 10.1016/j.compind.2021.103509]

20 Truong HT, Ta BP, Le QA, *et al*. Light-weight federated learning-based anomaly detection for time-series data in industrial control systems. Computers in Industry, 2022, 140: 103692. [doi: 10.1016/j.compind.2022.103692]

21 Kursawe K, Danezis G, Kohlweiss M. Privacy-friendly aggregation for the smart-grid. Proceedings of the 11th International Symposium on Privacy Enhancing Technologies. Cham: Springer, 2011. 175–191. [doi: 10.1007/978-3-642-22263-4_10]

22 Li XH, Cheng LX, Sun C, *et al*. Federated-learning-empowered collaborative data sharing for vehicular edge networks. IEEE Network, 2021, 35(3): 116–124. [doi: 10.1109/MNET.011.2000558]

23 Wang ZP, Dong NQ, Sun JH, *et al*. zkFL: Zero-knowledge proof-based gradient aggregation for federated learning. IEEE Transactions on Big Data, 2024. [doi: 10.1109/TBDATA.2024.3403370]

24 Choi B, Sohn J, Han DJ, *et al*. Communication-computation efficient secure aggregation for federated learning. Proceedings of the 9th International Conference on Learning Representations. Vienna: OpenReview.net, 2021.

25 Khojir HF, Alhadidi D, Rouhani S, *et al*. FedShare: Secure aggregation based on additive secret sharing in federated learning. Proceedings of the 27th International Database Engineered Applications Symposium. Heraklion: ACM, 2023. 25–33. [doi: 10.1145/3589462.3589504]

26 Lu Y, Yu ZX, Suri N. Privacy-preserving decentralized federated learning over time-varying communication graph. ACM Transactions on Privacy and Security, 2023, 26(3): 33. [doi: 10.1145/359135]

27 Shokri R, Stronati M, Song CZ, *et al*. Membership inference attacks against machine learning models. Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP). San Jose: IEEE, 2017. 3–18. [doi: 10.1109/SP.2017.41]

28 Fredrikson M, Jha S, Ristenpart T. Model inversion attacks that exploit confidence information and basic countermeasures. Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. Denver: ACM, 2015. 1322–1333. [doi: 10.1145/2810103.2813677]

29 Pereira D, Reis PR, Borges F. Secure aggregation protocol based on DC-nets and secret sharing for decentralized federated learning. Sensors, 2024, 24(4): 1299. [doi: 10.3390/s24041299]

30 Mugunthan V, Peraire-Bueno A, Kagal L. PrivacyFL: A simulator for privacy-preserving and secure federated learning. Proceedings of the 29th ACM International Conference on Information & Knowledge Management. ACM, 2020. 3085–3092. [doi: 10.1145/3340531.3412771]

31 Kanchan S, Jang JW, Yoon JY, *et al*. Efficient and privacy-preserving group signature for federated learning. Future Generation Computer Systems, 2023, 147: 93–106. [doi: 10.1016/j.future.2023.04.017]

32 Shamir A. How to share a secret. Communications of the ACM, 1979, 22(11): 612–613. [doi: 10.1145/359168.359176]

33 Boneh D, Gentry C, Lynn B, *et al*. Aggregate and verifiably encrypted signatures from bilinear maps. Proceedings of the 22nd International Conference on the Theory and Applications of Cryptographic Techniques. Warsaw: Springer, 2003. 416–432. [doi: 10.1007/3-540-39200-9_26]

34 Gordon Dr D. Discrete logarithm problem. In: van Tilborg HCA, Jajodia S, eds. Encyclopedia of Cryptography and Security. 2nd ed., Boston: Springer. 352–353. [doi: 10.1007/978-1-4419-5906-5_445]

35 Joux A, Nguyen K. Separating decision Diffie-Hellman from computational Diffie-Hellman in cryptographic groups. Journal of Cryptology, 2023, 16(4): 239–247. [doi: 10.1007/s00145-003-0052-4]

36 He KM, Zhang XY, Ren SQ, *et al*. Deep residual learning for image recognition. Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Las Vega: IEEE, 2016. 770–778.

37 Huang G, Liu Z, van der Maaten L, *et al*. Densely connected convolutional networks. Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Honolulu: IEEE, 2017. 4700–4708.

38 Hochreiter S, Schmidhuber J. Long short-term memory. Neural Computation, 1997, 9(8): 1735–1780. [doi: 10.1162/neco.1997.9.8.1735]

(校对责编: 张重毅)