基于动态信誉与 BLS 聚合驱动的改进型 PBFT 算法

成鹏辉[1,2] 范卓昆[1,2]
CHENG Penghui FAN Zhuokun

摘 要 实用拜占庭容错共识（practical byzantine fault tolerance, PBFT）算法因主节点随机选举、共识阶段需完成大量签名验证，存在显著的通信与计算开销。针对这一问题，文章提出融合动态信誉评分与 BLS 聚合签名的改进型共识算法——DB-PBFT（dynamic reputation and BLS-based PBFT）。该算法从准确性、稳定性、活跃度三个维度，对节点行为开展周期性量化评估，以此实现主节点的动态优选与轮换；同时引入 BLS（Boneh Lynn Shacham）聚合签名技术，在投票阶段完成签名的批量合并与一次性验证，大幅简化签名验证流程。实验结果表明，改进后的 DB-PBFT 算法有效降低了系统通信延迟，显著提升了共识效率，且在不同节点规模、高并发交易的场景中均具备良好的适应性。

关键词 区块链；共识算法；动态信誉评分；动态权值分配；BLS 聚合签名

doi：10.3969/j.issn.1672-9528.2026.02.002

0 引言

区块链技术作为一种去中心化的分布式账本技术，在金融、物联网、供应链等领域得到广泛应用。其内置的共识算法作为保障系统一致性与完整性的核心机制，备受关注[1-3]。其中，PBFT（practical byzantine fault tolerance）算法在众多算法中脱颖而出，得益于其在允许部分节点失效或存在恶意行为的前提下仍能达成一致的能力[4-6]。

然而，随着节点数量增多和大规模请求交易下，传统 PBFT 算法暴露出以下若干问题：首先，主节点的选举受固有机制影响缺乏弹性，无法根据节点历史与当前行为进行实时调整，降低了鲁棒性[7-8]；其次，PBFT 在投票阶段需收集、验证大量签名，带来显著的通信与计算开销[9]；此外，系统缺乏对节点能力的全面评估机制，极大地限制了对节点的动态管理[10]。

为解决上述出现的问题，本文提出一种融合动态信誉评分机制与 BLS 聚合签名驱动的改进型 PBFT 算法——DB-PBFT（dynamic reputation and BLS-based PBFT）。本算法在传统 PBFT 基础上引入三维度动态信誉评分体系，从而达到对节点行为进行实时量化评估，并依据信誉得分动态轮换主节点的目的，从根本上提升系统的弹性与安全性。同时，针对在投票阶段出现的反复签名与验证的问题，引入 BLS 聚合签名技术，只需聚合部分签名且统一验证，极大程度上降低了通信与计算开销，提高了共识效率。

1 相关工作

PBFT 作为一种经典的分布式共识算法，其核心优势在于可容忍最多 1/3 的恶意节点。然而，随着节点数量的增长，交易请求的增多，逐渐暴露出通信复杂度高、主节点选取机制单一等问题[6]。为应对上述挑战，文献 [11] 提出了一种改进算法 RB-PBFT，通过引入节点评估和惩罚机制以此抑制恶意行为，其虽然提升了系统的安全性和共识效率，但并未考虑大规模并发运行的网络问题。文献 [12] 基于 EigenTrust 模型提出了一种 T-PBFT 算法，通过交易行为为评估节点信任度，构建高质量共识组，提升了容错能力并降低通信成本。

此外，文献 [13] 提出的 FY-PBFT 算法通过声誉分数机制与 Fisher-Yates 洗牌算法动态评估节点表现，优化主节点选取；文献 [14] 的 NR-PBFT 算法则基于可信度动态划分共识节点集群，进一步降低了主节点被恶意操控的风险。然而，以上研究在大规模节点环境下仍面临通信负担重、系统负载高等问题。

为进一步优化 PBFT 的性能与安全性，研究者引入了 BLS（Boneh Lynn Shacham）聚合签名算法[15]。文献 [16] 提出的 ABFT 结合 BLS 签名显著降低了网络传输与验证开销，但主节点选取机制仍待改进。因此，本文提出一种 DB-PBFT 的优化算法，旨在兼顾共识效率、节点选举公平性与系统通信负载之间的平衡。

1. 太原师范学院计算机科学与技术学院 山西晋中 030619
2. 太原师范学院智能优化计算与区块链技术山西省重点实验室 山西晋中 030619

[基金项目] 太原师范学院研究生教育创新项目（SYYJSYC-2479）；太原师范学院研究生教育创新项目（SYYJSYC-2480）

2 DB-PBFT 算法设计

2.1 设计思想

DB-PBFT 共识算法在传统 PBFT 协议基础上进行了系统性的结构优化和机制重构，核心目标是增强其在动态网络环境下的自适应能力、共识效率与系统鲁棒性。从整体上看，DB-PBFT 设计逻辑主要包括三个方面：

（1）在节点行为层面：构建三维度动态信誉评分体系，其中，节点的历史响应是否正确、是否参与所有阶段、是否频繁掉线等信息都被纳入信誉评分系统，作为主节点选举的重要依据。

（2）在消息传播层面：引入 BLS 签名对投票阶段进行改进，包括 Prepare 和 Commit 阶段各节点的签名进行聚合验证。

（3）在系统控制层面：设计统一的状态同步机制，确保参与的所有节点在广播前均完成监听与注册，以此保证共识流程的完整进行。

DB-PBFT 的整体结构围绕“信誉驱动 + 聚合签名 + 流程同步”三大核心理念展开，各模块之间紧密协作，形成了完整的共识执行闭环。算法执行过程中，信誉驱动主节点选择，主节点控制共识流程广播，流程完成后更新各节点信誉值并判断是否轮换，最终构建出动态可信、高效执行、适应变化的共识机制。

2.2 系统模块

本系统将共识节点划分为 6 个模块，各司其职、协同运行：（1）共识模块负责处理客户端请求，完成 Pre-Prepare、Prepare 和 Commit 阶段消息交互；（2）信誉模块基于节点行为动态更新评分 DRI（dynamic reputation index），为主节点轮换提供依据；（3）轮换模块依据评分结果定期发起视图切换；（4）签名模块实现 BLS 签名与聚合验证，减少通信开销；（5）日志模块记录共识状态与异常信息，便于故障恢复；（6）网络模块维护节点间连接，支持高效广播。

各模块通过共享状态表与回调机制实现解耦通信，当完成聚合验证后，将信号发送至共识模块触发状态推进；当 DRI 模块检测到信誉低下节点时，自动提交惩罚请求至轮换模块。上述机制构成了一个稳定、高效且可演化的共识节点内部协同体系。如图 1 所示。

2.3 DB-PBFT 共识流程

DB-PBFT 共识流程是在传统 PBFT 协议的五阶段结构（Request、Pre-Prepare、Prepare、Commit、Reply）基础上进行优化与重构，引入了 DRI 与 BLS 聚合签名机制，降低通信与验证负载，提升系统在大规模动态环境下的效率与鲁棒性，具体流程如图 2 所示。

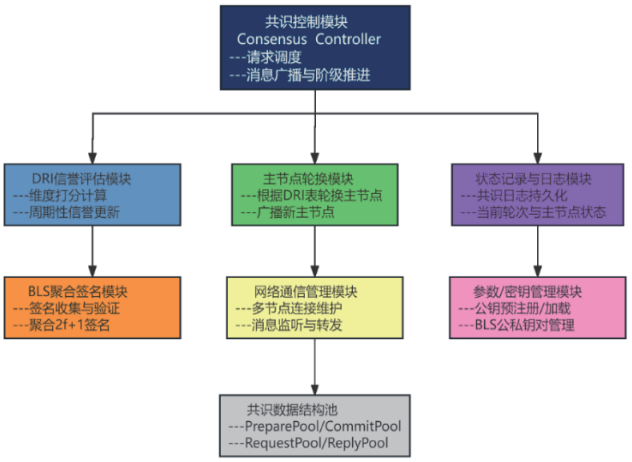

图 1 整体结构图

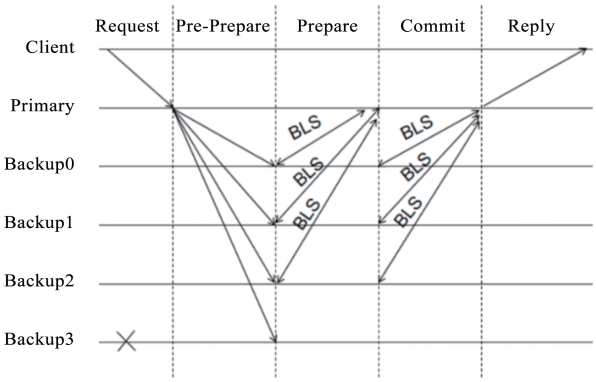

图 2 DB-PBFT 共识流程

Request 阶段：客户端构造交易请求消息，内容包含请求编号、时间戳、操作内容及客户端签名，随后发送至当前视图的主节点。

Pre-Prepare 阶段：主节点接收到客户端请求后，封装生成 Pre-Prepare 消息，包含：请求摘要、视图编号、请求编号等信息，并广播至所有副本节点。

Prepare 阶段：副本节点收到 Pre-Prepare 消息并验证通过后，生成本节点的 Prepare 签名消息并发送给主节点（聚合节点）。主节点在收到来自其他副本的签名后，聚合至少 $2f+1$ 个合法签名并将结果广播至所有副本节点，副本节点收到聚合签名后进行一次验证，验证通过则进入提交阶段。

Commit 阶段：各副本节点在验证通过 Prepare 聚合签名后，立即生成本节点的 Commit 签名并发送至主节点。主节点同样收集 $2f+1$ 个 Commit 签名，执行聚合，并广播聚合签名至所有副本。

Reply 阶段：共识达成后，节点执行交易操作并将结果返回给客户端。在实现层面，为降低客户端处理负担，采用主节点统一回复策略。

3 DB-PBFT 核心机制设计

3.1 DRI 动态信誉评分机制设计

在 DB-PBFT 算法中，DRI 用于量化各共识节点在系统运行过程中的表现，并据此动态调整主节点选取策略。该机制在 PBFT 基础上引入节点行为反馈调节，通过周期性更新信誉值实现主节点弹性轮换，为系统提供基于历史行为数据的角色决策依据。

每个节点 i 的信誉值 R_i 由准确性$R_{\text{acc}}^{(i)}$、稳定性$R_{\text{sta}}^{(i)}$与活跃度$R_{\text{act}}^{(i)}$三个维度指标加权组成，综合评分公式为：

$$R_i = \alpha \cdot R_{\text{acc}}^{(i)} + \beta \cdot R_{\text{sta}}^{(i)} + \gamma \cdot R_{\text{act}}^{(i)} \tag{1}$$

式中：α、β、γ分别表示三个维度的加权系数，满足$\alpha+\beta+\gamma=1$；$R_{\text{acc}}^{(i)}$表示节点响应消息的正确率，定义为$R_{\text{acc}}^{(i)} = N_{\text{valid}}^{(i)} / N_{\text{total}}^{(i)}$，其中$N_{\text{valid}}^{(i)}$表示节点被其他节点确认有效的响应次数，$N_{\text{total}}^{(i)}$表示节点总响应次数；$R_{\text{sta}}^{(i)}$表示衡量节点在历史共识过程中的稳定性，定义为 $R_{\text{sta}}^{(i)} = P_{\text{complete}}^{(i)} / P_{\text{total}}^{(i)}$，其中$P_{\text{complete}}^{(i)}$表示节点成功完成的共识阶段数，$P_{\text{total}}^{(i)}$表示其参与的总阶段数；$R_{\text{act}}^{(i)}$表示节点参与度，定义为$R_{\text{act}}^{(i)} = T_{\text{active}}^{(i)} / T_{\text{window}}^{(i)}$，其中$T_{\text{active}}^{(i)}$表示节点在当前窗口$T_{\text{window}}^{(i)}$内的活跃时长。

最后，在每轮共识完成后，系统会收集各节点在当前轮次中的行为数据，并更新其在动态信誉索引表（DRITable）中的信誉分数。为了避免偶然情况导致的节点某次分数突增，本文采用滑动加权平均策略对上一轮得分和本轮行为进行处理。具体计算方式为：

$$R_i^{(t)} = \lambda \cdot R_i^{(t-1)} + (1-\lambda) \cdot R_i^{(\text{current})} \tag{2}$$

式中：$\lambda \in [0,1]$。

3.2 BLS 聚合签名机制设计

BLS 签名算法基于双线性对（bilinear pairing）构造，能够将多个签名压缩为一个固定长度的聚合签名，并通过一次性验证确认其合法性，从而显著提升共识效率。

3.2.1 BLS 签名基础

设定 G_1、G_2 为 q 阶的椭圆曲线群，e: G_1、$G_2 \to G_T$ 为双线性映射；$g \in G_1$，哈希函数 H: $\{0,1\}^* \to G_1$ 将消息映射到 G_1 上的点；每个点 i 的私钥和公钥为$\text{sk}_i \in \mathbb{Z}_q$，$\text{pk}_i = g^{\text{sk}_i} \in G_1$。BLS 签名过程如下：

对消息 m，节点 i 计算签名为$\sigma_i = H(m)^{\text{sk}_i}$，之后检查$\text{e}(\sigma_i, g) = \text{e}(H(m), \text{pk}_i)$是否成立。

3.2.2 Prepare 与 Commit 阶段聚合策略

在 DB-PBFT 中，当节点收到来自其他 $2f+1$ 个节点的消息 m 并验证其合法性，之后将其单签名$\sigma_i = H(m)^{\text{sk}_i}$进行聚合处理。设 $S = \{\sigma_1, \sigma_2, \cdots, \sigma_k\}, k \geq 2f+1$ 是一组对 m 的有效 BLS 签名，其聚合签名为$\sigma_{\text{agg}} = \prod_{i=1}^{k} \sigma_i = H(m)^{\sum_{i=1}^{k} \text{sk}_i}$，统一广播 <PREPARE/COMMIT, M, σ_{agg}, $\{\text{pk}_i\}$>，其他节点接收后进行一次聚合验证，确认消息合法性。验证过程为：

$$\text{e}(\sigma_{\text{agg}}, g) = \prod_{i=1}^{k} \text{e}(H(m), \text{pk}_i) \tag{3}$$

具体 BLS 聚合签名处理流程伪代码如算法 1 所示。

算法 1 BLS 聚合签名处理

输入：节点 N_i，摘要 digest 输出：聚合结果

```
1. IF isPrimary(N_i) THEN
2.    σ_i ← Sign(digest, sk_i)
3.    Broadcast <PRE-PREPARE, digest, σ_i>
4. ELSE
5.    Wait for <PRE-PREPARE, digest, σ_p>
6.    IF Verify(digest, σp, pkp) = true THEN
7.       σ_i ← Sign(digest, sk_i)
8.       Broadcast <PREPARE, digest, σ_i>
9.    END IF
10. END IF
11. Wait for PREPARE messages from at least 2f+1 nodes
12. IF All σ_j pass Verify THEN
13.    σ_agg ← Aggregate({σ_j})
14.    Broadcast <COMMIT, digest, σ_agg>
15. END IF
16. Wait for COMMIT messages from at least 2f+1 nodes
17. IF Aggregated signature σagg passes Verify THEN
18.    Execute transaction
19. END IF
```

3.3 主节点动态轮换机制设计

DB-PBFT 主节点轮换机制不再依赖固定时间窗口或轮数，而是依据以下两类事件驱动轮换：

（1）周期性评估轮换：在每轮共识完成后，系统重新评估全网节点的 DRI 分数，所有节点根据最新的 DRITable 从中选取当前信誉值最大的节点作为主节点。

（2）故障容忍触发： 若当前主节点连续丢失共识阶段（如未完成 Pre-Prepare 消息广播），被 $2f+1$ 个副本节点联合上报，将立即剥夺其主节点资格并轮换。

所有节点维护一张最新的 DRITable=(ID_i, R_i)，排序获得信誉最高的节点索引 $\text{ID}_{\max}$。设当前视图编号为 v，主节点为$\text{ID}_{\text{primary}}^{(v)}$，若$\text{ID}_{\max} \neq \text{ID}_{\text{primary}}^{(v)}$且$R_{\max} > R_{\text{threshold}}$则进入视图切换流程，设置新主节点为$\text{ID}_{\text{primary}}^{(v+1)} = \text{ID}_{\max}$，所有节点通过广播接收同步后的主节点视图，具体如算法 2 所示。

算法 2 主节点轮换机制

输入：当前主节点参数 输出：更新后的主节点编号

```
1. IF (current_time - last_rotation_time < T_cooldown) THEN
2.     RETURN current_primary
3. END IF
4. reputation_list ← sort_by_reputation(DRITable)
5. new_primary ← reputation_list[0]
6. IF new_primary ≠ current_primary THEN
7.     view_number ← view_number + 1
8.     broadcast_view_change(view_number, new_primary)
9.     update_primary_log(view_number, current_primary, new_
primary, current_time)
10.    last_rotation_time ← current_time
11.    RETURN new_primary
12. ELSE
13.    RETURN current_primary
14. END IF
```

4 实验分析

4.1 实验环境

为全面评估 DB-PBFT 共识机制在性能的改进效果，本文基于自主实现的原型系统进行实验部署，实验软硬件配置信息如表 1 所示。

表 1 实验配置信息

对象	配置
实验平台	Ubuntu 22.04 LTS 64-bit
处理器	Intel Core i7-12700K
内存	32 GB DDR4
容器化部署	Docker 28.0.1
BLS 签名库	herumi/bls-eth-go-binary
编程语言	Go 1.23.6

4.2 实验结果分析

为验证所提 DB-PBFT 共识算法在处理性能上的优势，本文将其与韩璐等 [14] 提出的 NRPBFT 算法在单次请求平均时延上进行了对比，如表 2 所示。由于 NRPBFT 未提供大规模请求下的数据，本文仅对 DB-PBFT 进行了高并发测试。实验中对多次请求的响应时间取平均，以 ms 为单位衡量性能。如图 3 所示，DB-PBFT 在各节点规模下平均时延均优于 NRPBFT。以 20 个节点为例，DB-PBFT 的平均时延为 8.30 ms，而 NRPBFT 为 36.87 ms，性能提升约 4.4 倍。尽管节点数量增加会提高共识复杂度，但得益于 BLS 聚合签名与动态信誉选主机制，有效减少了通信开销与签名验证延迟，使 DB-PBFT 在复杂高并发环境下仍保持较低时延。

表 2 单次请求时延对比

节点数	NRPBFT	DB-PBFT
4	10.94	1.5
6	10.70	2.2
8	11.96	3.1
10	13.01	3.4
12	14.96	4.1
14	15.95	4.4
16	20.90	5.3
18	30.92	6.4
20	36.87	8.3

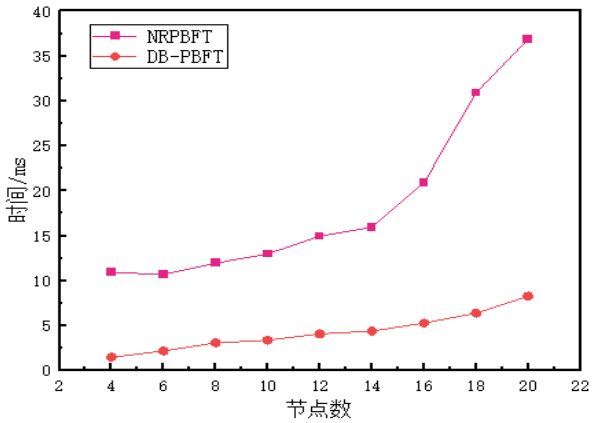

图 3 对比方案

另外为进一步测试 DB-PBFT 算法在高频交易场景中的处理能力，本文设计了 100 次连续请求的实验，记录在不同节点数配置下的平均响应时间与系统吞吐量如表 3 和图 4 所示。

表 3 100 次请求实验数据

节点数	平均时延	吞吐量
4	3.3	334.36
6	7.2	138.24
8	17.3	57.91
10	32.8	30.47
12	37.8	26.45
14	126.0	7.94
16	200.8	4.98
18	359.7	2.78
20	513.7	1.95

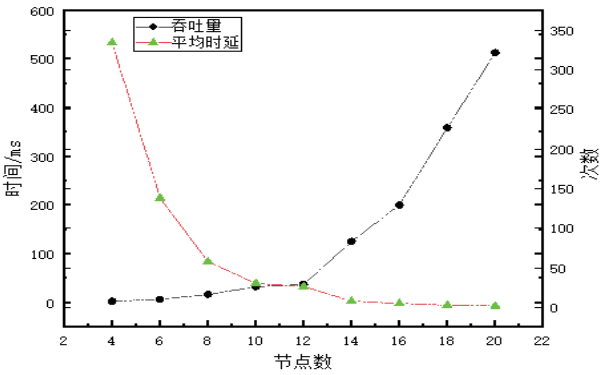

图 4 100 次请求实验

综合实验结果可得出，在低负载场景中，DB-PBFT 的响应性能与现有改进算法相当，具备毫秒级甚至亚毫秒级处理能力；在高并发、多轮请求场景中，DB-PBFT 显示出更优的可扩展性和稳定性，尤其在中等节点规模（5~8 个节点）下仍能保持较高的处理效率。

5 结论

本文围绕 PBFT 当前存在的性能瓶颈与安全隐患，提出了一种 DB-PBFT 算法，并实现了完整系统架构与验证实验。相比现有改进方案，该算法不仅实现了弹性主节点轮换，而且降低了通信与计算开销，具备良好的扩展性与适应性。然而，当前在聚合签名过程中对网络同步和消息完整性存在依赖，系统在异步环境下的鲁棒性尚待验证。未来研究可进一步拓展算法在跨链共识、实际联盟链部署中的应用潜力。

参考文献：

[1]WU H J, YAO Q, LIU Z G, et al. Blockchain for finance: a survey[J]. IET blockchain, 2024, 4(2): 101-123.

[2]ENAYA A, FERNANDO X N, KASHEF R. Survey of blockchain-based applications for IoT[J]. Applied sciences, 2025, 15(8): 4562.

[3]LUO Z D. A survey on blockchain-based supply chain finance with progress and future directions[EB/OL].(2024-08-14)[2025-12-30].https://doi.org/10.48550/arXiv.2408.08915.

[4]CASTRO M, LISKOV B. Practical byzantine fault tolerance[C]//3rd Symposium on Operating Systems Design and Implementation(OSDI 1999). NewYork: ACM, 1999: 173-186.

[5]TANG S, WANG Z Q, JIANG J, et al. Improved PBFT algorithm for high-frequency trading scenarios of alliance blockchain[J]. Scientific reports, 2022, 12: 4426.

[6]YUAN F J, HUANG X, ZHENG L, et al. The evolution and optimization strategies of a PBFT consensus algorithm for consortium blockchains[J]. Information, 2025, 16(4): 268.

[7]LIU S N, ZHANG R H, LIU C Z, et al. An improved PBFT consensus algorithm based on grouping and credit grading[J]. Scientific reports, 2023, 13(1): 13030.

[8]DING J W, WU X, TIAN J, et al. RE-BPFT: an improved PBFT consensus algorithm for consortium blockchain based on node credibility and ID3-based classification[J]. Applied sciences, 2025, 15(13): 7591.

[9]XIAO J, LUO T, LI C Q, et al. CE-PBFT: a high availability consensus algorithm for large-scale consortium blockchain[J]. Journal of king saud university-computer and information sciences, 2024, 36(2): 101957.

[10]WANG Y, WAN Q C, WU Y F, et al. Grouped byzantine fault tolerant consensus algorithm based on aggregated signatures[J/OL].Cybersecurity,2025[2025-12-30].https://link.springer.com/article/10.1186/s42400-025-00362-9.

[11]HE F Y, FENG W L, ZHANG Y, et al. An improved byzantine fault-tolerant algorithm based on reputation model[J]. Electronics, 2023, 12(9): 2049.

[12]GAO S, YU T Y, ZHU J M, et al. T-PBFT:an eigentrust-based practical byzantine fault tolerance consensus algorithm[J]. China communications,2019,16(12):111-123.

[13] 徐莹臣，田野，景莹，等．基于声誉分数的改进 PBFT 共识算法 [J]. 信息技术与信息化，2025(3):179-183.

[14] 韩璐，翟健宏，杨茹．基于节点可信度的改进 PBFT 算法 [J]. 智能计算机与应用，2023, 13(7):191-201.

[15]LONG J Y, WEI R B. Scalable BFT consensus mechanism through aggregated signature gossip[C]//2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC). Piscataway: IEEE, 2019: 360-367.

[16]KNUDSEN H, LI J Y, NOTLAND J S, et al. High-performance asynchronous byzantine fault tolerance consensus protocol[C]//2021 IEEE International Conference on Blockchain (Blockchain). Piscataway: IEEE, 2021: 476-483.

【作者简介】

成鹏辉（2000—），男，山西吕梁人，硕士研究生，研究方向：区块链、属性基加密，email:17836056180@163.com。

范卓昆（2000—），男，山西运城人，硕士研究生，研究方向：区块链、属性基加密。

（收稿日期：2025-07-30 修回日期：2026-02-02）