Software Engineering Group
Department of Computer Science
Nanjing University
http://seg.nju.edu.cn

Technical Report No. NJU-SEG-2025-IJ-001

2025-IJ-001

BlockSOP: A blockchain-based software management platform for open collaborative development

Shuoxiao Zhang, Enyi Tang, Haoliang Cheng, Xinyu Gao, An Guo,

Jianhua Zhao, Xin Chen, Linzhang Wang, Na Meng, Xuandong Li

Technical Report 2025

The Journal of Systems and Software 230 (2025) 112477

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

BlockSOP: A blockchain-based software management platform for open collaborative development☆

Shuoxiao Zhang [a], Enyi Tang [a,*], Haoliang Cheng [a], Xinyu Gao [a], An Guo [a], Jianhua Zhao [a], Xin Chen [a], Linzhang Wang [a], Na Meng [b], Xuandong Li [a]

[a] State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, 210093, China
[b] Virginia Polytechnic Institute and State University, Blacksburg VA 24060, USA

ARTICLE INFO

Keywords:
Collaborative development
Contribution
Incentive
Motivation
Blockchain
Smart contract

ABSTRACT

Open collaborative development is common and brings tremendous benefits to developers and users. However, it also poses new challenges. Incentives are a crucial mechanism for promoting open collaborative development, but developers find it difficult to receive rewards corresponding to their contributions. This has prompted us to propose a trustworthy incentivization mechanism for collaborative software development based on blockchain (*BlockSOP*) to enhance user development motivation. We have innovatively introduced the concept of *contribution points* to measure users' contributions to collaborative development. Smart contracts fairly allocate project earnings based on the value of users' *contribution points*. Additionally, through smart contracts, we have implemented a secure mechanism combining *deposits* and *credit scores* to effectively reduce the risks of software information leakage and malicious behavior during the collaborative development process. Experimental results demonstrate that *BlockSOP* effectively safeguards developers' interests and significantly enhances developers' development motivation and response speed while keeping expenses within a reasonable range.

1. Introduction

Open collaboration has emerged as a critical component of modern software development practices, affording numerous benefits to developers and users (Constantino et al., 2023; Barcomb et al., 2020). By enabling the sharing of knowledge and expertise across a broad, open network, open-source development can substantially enhance the efficiency and quality of software development (Tan et al., 2023), facilitating greater innovation and fostering the creation of software products (Assavakamhaenghan et al., 2022; Samuel et al., 2022).

However, how to effectively motivate developers to contribute is a major challenge for open source development, which has led to the demise of many potential projects (Howison and Herbsleb, 2013; Qiu et al., 2019a; Dijkers et al., 2018). Unlike traditional proprietary software development models that rely on profit margins and licensing fees, open-source projects often rely on donations, grants, or volunteer contributions, which can be unpredictable and unreliable (Shimada et al., 2022; Zhang et al., 2022). The challenge of sustaining open-source projects is compounded by the fact that developers often prioritize their paid work over their contributions to open-source projects (Zhang et al., 2024; Barcomb et al., 2022, 2020; Jamieson et al., 2024), resulting in a lack of consistent and reliable development resources (Coelho and Valente, 2017; Iaffaldano et al., 2019b; Li et al., 2022). Additionally, while open-source projects often have strong communities that support their development, these communities may not be able to provide the incentives needed to sustain projects in the long run (Qiu et al., 2019b; Curto-Millet and Jiménez, 2023; Valiev et al., 2018a).

These challenges have led researchers and practitioners to explore alternative funding models for open-source development, such as crowdfunding (Zhou et al., 2021), corporate sponsorship (Butler et al., 2021), and grant funding (Overney et al., 2020; Overney, 2020). While these models can provide some level of incentives (Nakasai et al., 2017; Fan et al., 2024a), they also introduce new challenges. In particular, developers often lack trust in each other (Wermke et al., 2022; Ho and Richardson, 2013; Moe and Šmite, 2008), and ensuring corresponding rewards for their contributions becomes difficult (Hann et al., 2013), resulting in developer attrition (Yu et al., 2012). In

☆ Editor: Tao Zhang.

* Corresponding author.
E-mail addresses: sx_zhang@smail.nju.edu.cn (S. Zhang), eytang@nju.edu.cn (E. Tang), dg1932001@smail.nju.edu.cn (H. Cheng), xinyugao@smail.nju.edu.cn (X. Gao), guoan218@smail.nju.edu.cn (A. Guo), zhaojh@nju.edu.cn (J. Zhao), chenxin@nju.edu.cn (X. Chen), lzwang@nju.edu.cn (L. Wang), nm8247@cs.vt.edu (N. Meng), lxd@nju.edu.cn (X. Li).

https://doi.org/10.1016/j.jss.2025.112477
Received 21 August 2024; Received in revised form 10 January 2025; Accepted 24 April 2025
Available online 23 May 2025

Table 1
A comparative analysis of features in BlockSOP, BountySource, Gitcoin and OpenCollective

Feature description	BlockSOP	BountySource	Gitcoin	OpenCollective
Ability to fairly distribute income to all involved contributors	√	×	×	×
Detection of malicious behaviors	√	×	√	√
Security mechanism to prevent copyright infringement	√	×	×	×
Direct investment income from projects	√	×	×	×
Progressively granting access rights of projects with fine-grained permissions	√	×	×	×
Transparent processes empowered by blockchain	√	×	√	×

financial services, trust is a relatively important factor influencing customer behavior (Tyler and Stanley, 2007; Pi et al., 2012; Leblang et al., 2022; Oláh et al., 2021). The lack of trust in open source collaboration can give rise to a series of issues, particularly in terms of code loopholes and inequitable incentive distribution. On one hand, distrust in open source collaboration can lead developers to focus on superficial outcomes, neglecting code quality and security, which in turn may introduce code loopholes, such as the infamous Heartbleed bug (Zhang et al., 2014). On the other hand, the lack of trust in incentive allocation is another critical issue facing the open source community. When incentive mechanisms fail to allocate resources fairly, some contributors' efforts may be overlooked (Lamprecht et al., 2020). For example, many open source developers on GitHub rely on unstable donations from sponsors, leading to a significant loss of contributors.

In light of these issues, this paper proposes *BlockSOP*, a novel blockchain-based platform for managing software development designed to enhance user motivation towards open collaboration in software projects. We define the *end-users* as true appreciators of the intrinsic value of the software and users willing to invest in the software. *BlockSOP* establishes a direct link between the principal source of revenue and the *end-users*. Through *progressive opening model* we proposed, developers and *end-users* create a mutually beneficial trading relationship with clearly defined profit margins. More details about the *progressive opening model* will be discussed in Section 2. Contributors of the project are rewarded with market income proportionate to their level of contribution. To ensure transparency and trustworthiness, we implement *BlockSOP* using blockchain techniques.

The *progressive opening model* forms the fundamental basis of *BlockSOP* and serves to safeguard the fine-grained proprietary rights of each contributor during the open collaborative software development process. This model involves granting developers from the open world progressively increasing access to the artifacts in a project based on their contributions. As research suggests, developers' intention to engage in copyright infringements decreases as their stake in the proprietary rights of the project increases (Lazear and Rosen, 1981). Hence, *BlockSOP* grants these developers increased access to the artifacts during open collaboration. Conversely, developers who lack sufficient proprietary rights may need to rent access rights in order to complete jobs in the open collaboration. They can pay back the borrowed access rights later by earning *contribution points* through their work on the job.

BlockSOP offers a novel approach to incentivize continued contributions to open collaborative software projects by rewarding project contributors with income from the market based on their percentage of contribution. To measure each contributor's proprietary stake in a project, we introduce the concept of *contribution points*. These points represent the proportion of proprietary rights a contributor has earned in a project and are determined by a predefined contribution point earning scheme specific to each project. Income from the market is then distributed among contributors based on the proportion of *contribution points* that they have earned. It is worth noting that different projects are independent of each other, and *contribution points* from one project cannot be used interchangeably with those of another project.

We introduce blockchain-based techniques to construct mechanisms for our models and schemes with transparency and trust. By leveraging smart contracts, *BlockSOP* automatically calculates and distributes income to contributors in accordance with their *contribution points*, eliminating the need for manual and potentially biased distribution methods. Moreover, the blockchain-based framework provides a tamper-proof record that ensures contributors' proprietary rights are protected throughout the software development process.

Furthermore, we have introduced blockchain-based technology to establish models and security mechanisms that are transparent and trustworthy. By deploying smart contracts, *BlockSOP* automatically calculates and distributes income to contributors based on their *contribution points*, eliminating potential biases that manual income distribution may bring. Additionally, the blockchain-based security mechanisms protect the proprietary rights of contributors, enhancing the trust of the entire development process.

The modular design of *BlockSOP* enhances both the system's efficiency and scalability, while optimizing computational complexity. Specifically, the Contribution Point Usage Module, with a constant complexity of $O(1)$, ensures high efficiency. The Get Contribution Point Module and Arbitration Module exhibit linear complexities of $O(N)$ and $O(M log M + K)$, respectively, effectively handling the growth in user scale. Although the watermark embedding and extraction modules are influenced by code size, their complexity of $O(L^2)$ remains relatively manageable. The overall complexity distribution of *BlockSOP* supports large-scale collaborative development, ensuring both operational efficiency and scalability.

In Table 1, we present a comparative analysis of *BlockSOP*, BountySource (Zhou et al., 2021a), Gitcoin (Choetkiertikul et al., 2023; Schmid and Shestakov, 2024) and OpenCollective (Zhou et al., 2022), highlighting their distinct features and functionalities. BountySource and Gitcoin represent pioneering platforms in the domain of software development crowdfunding. BountySource primarily offers rewards for achieving specific tasks or goals, while Gitcoin harnesses the potential of blockchain technology to facilitate funding for open-source development. OpenCollective is a platform designed to facilitate transparent and collaborative funding for open-source projects and community-driven initiatives. However, these platforms exhibit limitations in motivating sustained contributions to open collaborative software projects, a formidable challenge in its own right, as well as ensuring fairness.

BlockSOP's advantage over BountySource, Gitcoin and OpenCollective in income distribution lies in its fairness based on smart contracts. Smart contracts automatically allocate rewards according to actual contributions, ensuring transparency and fairness in income distribution. In contrast, BountySource, Gitcoin and OpenCollective rely on human judgment, which is susceptible to subjective decisions or external factors, leading to potential inequities. *BlockSOP*'s malicious behavior detection mechanism is supported by blockchain's security features, enabling the timely identification and tracking of malicious activities. In comparison, Gitcoin and OpenCollective primarily depends on community feedback and manual review, which introduces delays and vulnerabilities, while BountySource lacks an effective malicious behavior detection system. As a result, *BlockSOP* is more efficient in preventing malicious actions. *BlockSOP* enhances copyright protection for projects through blockchain technology, ensuring that code ownership and copyrights can be verified and traced via on-chain data. In contrast, BountySource, Gitcoin and OpenCollective lack a systematic approach to copyright protection. By integrating blockchain

smart contracts, *BlockSOP* ensures transparent fund flow, with investor funds directly invested into projects, generating returns automatically based on project progress. In comparison, BountySource, Gitcoin and OpenCollective primarily rely on funding mechanisms, which have a more indirect and opaque investment return structure. Thus, *BlockSOP* offers investors a clearer and more direct path to returns. In terms of access control, *BlockSOP* uses a progressive opening model to ensure that sensitive information is only accessible to authorized personnel, a level of fine-grained management that is difficult to achieve in BountySource, Gitcoin and OpenCollective. *BlockSOP* provides a public and transparent workflow based on blockchain, ensuring transparency in project management, fund flow, and code submissions. In contrast, BountySource and OpenCollective still depend on backend management, lacking transparency.

The main contributions of this paper are as follows:

- We propose a blockchain-based trustworthy management platform for collaborative software development, with the aim of enhancing users' motivation for development contributions. Innovatively, we introduce the concept of shares in software collaboration, distributing software project earnings based on user contribution percentages, while ensuring users' rightful interests through blockchain technology.
- We establish a security mechanism based on a combination of deposit and credit scores to prevent malicious behavior and code leakage during collaborative development. Using game theory, we simulate user profits under different decision scenarios and conduct user surveys to demonstrate the validity of the security mechanism and the trustworthiness of the entire development process.
- By comparing experimental data with BountySource and OpenCollective, we have demonstrated that *BlockSOP* is superior to BountySource in enhancing software development motivation and response speed, *BlockSOP* is more likely to attract investors' attention than OpenCollective.
- We experimentally test the gas consumption and performance of *BlockSOP* and Gitcoin, and the results indicate that *BlockSOP* outperforms Gitcoin slightly in terms of throughput and latency, and the overhead is also within a reasonable range.

The remaining sections of the paper are structured as follows: In Section 2, we provide a detailed explanation of the *progressive openness model*. Subsequently, Section 3 introduces the background of blockchain tamper-proof and smart contracts. Moving forward, Section 4 presents the key modules of our mechanism along with their implementation details. Our security mechanism is elaborated upon in Section 5. The evaluation results and corresponding discussion are presented in Section 6. Section 7 presents a comprehensive overview of the related work pertaining to this paper. Lastly, in Section 8, we conclude our findings and discuss future work.

2. Progressive opening model

Our blockchain-based software development management platform, *BlockSOP*, endeavors to improve the software development landscape by fostering transparency and sustainability. This innovative paradigm is underpinned by the conviction that valuable open-source software especially need *end-users* financing (Bonaccorsi and Rossi, 2003). Traditional open-source financing models, dependent on sponsor support, can inadvertently undermine the customer experience, as software features cater predominantly to sponsor interests (Zhou et al., 2021b; Wang et al., 2022). In certain instances of sponsored open-source software, sponsors may view the software products as tools serving their corporate strategies, rather than valuable resources that address customer needs (Fan et al., 2024b). Recognizing that the software development industry is driven by innovation and customer requirements (Jahn et al., 2024; Zahedi et al., 2018; Casadesus-Masanell and Llanes, 2015), we posit that a customer-centric approach fosters enhanced satisfaction, positive word-of-mouth, and increased product adoption, thereby nurturing a virtuous cycle of developmental productivity.

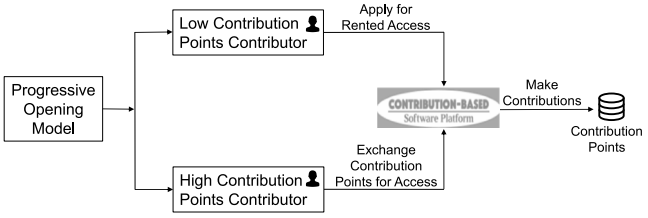


Fig. 1. The process through which different contributors gain deep access to the project via the progressive opening model.

Achieving the dual goals of openness and financial viability presents a formidable challenge (Voong and Saebi, 2023). To address this challenge, we propose an innovative solution: the *progressive opening model*. This model seeks to optimize collaborative efforts by striking a balance between these competing interests. It offers a practicable financing framework that implements a permission-based system, restricting access to software components based on a contributor's role. By restricting interactions to the only components necessary for a given contribution, the *progressive opening model* mitigates potential project leakage, thus ensuring the integrity of the project and the feasibility of financing strategies. Simultaneously, the model reduces the risk of information leaks from high-access roles by directly associating access levels with project ownership and tying ownership to the project's benefits.

As illustrated in Fig. 1, contributors in *BlockSOP* can be categorized into two types: high contribution points contributor and low contribution points contributor. The progressive open model assigns different levels of access based on the contributor's degree of involvement. Low contribution points contributor can rent contribution points to gain access to specific project details, thereby enhancing overall understanding of the project. Upon completing his contributions, he is required to return the corresponding rented points. This rental mechanism ensures that low contribution points contributor can acquire deep understanding of the project. In contrast, high contribution points contributor can directly exchange contribution points for access to the entire project. These two options provide flexible choices for contributors with varying levels of involvement, ultimately improving the efficiency and effectiveness of collaborative development.

In particular, this model operates on the principle of ownership allocation commensurate with individual contribution. Greater contributions to the software project yield more substantial ownership rights. These rights confer three key privileges: a larger share of project benefits, decision-making power on significant project matters, and access to a broader range of core and peripheral project components. This strategic amalgamation of rights serves to deter key contributors from disclosing sensitive project information, thereby addressing potential trust issues during the open collaboration process. High-access roles, by virtue of their substantial stake in the project's benefits, are less likely to jeopardize their own interests by leaking valuable project assets.

In open collaborations involving ordinary contributors with limited ownership rights, these contributors only necessitate access to components essential for their respective tasks. In order to inherit the spirit of open source, the *progressive opening model* introduces a mechanism to allow *'rented'* access to newcomers, in which novice contributors need to demonstrate the value of their contributions to project decision-makers and prove that the access permissions they request are reasonable in order to successfully obtain access permissions. *Project owners* are composed of the project creators, outstanding developers, and excellent investors. They will consider the validity of the contribution and the

extent of access requested, assess the risk and make a final decision to grant or withhold access. This decision-making process involves voting among multiple decision-makers, emphasizing the *project owners*' critical role, as their decisions impact the interests of larger ownership roles.

The model also provides for *rented* access to be reimbursed by contributors through *contribution points*, awarded upon task completion. We refer to the *contribution points* that have not yet exchanged code access permissions as *unexecuted contribution points* and the *contribution points* that have exchanged code access permissions as *executed contribution points*. These points have inherent access rights governed by our platform's rules. Initially, *contribution points* are unexecuted and can be exchanged for specific code access rights. After this exchange, the points become executed and can only be used to obtain a share in software revenue and decision-making rights. Consequently, novice contributors can reimburse their '*rented*' access by converting their *contribution points* from unexecuted to executed status. Meanwhile, established contributors can acquire access rights for new tasks by redeeming previously accrued *unexecuted contribution points*. This model addresses insufficient access rights while preserving project security and financial viability in open collaboration.

3. Background

In this section, we briefly present key background information about blockchain *tamper-proof* and *smart contracts*.

The blockchain is fundamentally a decentralized, distributed peer-to-peer (P2P) network wherein all nodes share equal status (Zantalis et al., 2024; Pacheco et al., 2023). This structure effectively mitigates network attacks targeting central nodes and addresses the issue of single-point failures. It accomplishes this by utilizing transactions as a medium for data communication and resource exchange (Monrat et al., 2019; Alzoubi et al., 2022). Serving as a unique form of database, the blockchain not only stores data but also ensures its trusted transmission. In practical applications, valuable assets can be digitized and securely stored within the blockchain (Zhou et al., 2024). Leveraging its inherent properties of tamper resistance, decentralization, and traceability, the blockchain can provide low-risk, cost-effective security service solutions (Ren et al., 2023).

The tamper-evident property of blockchain arises from its unique structure: blocks, containing transaction data, are chronologically added to the chain (Nguyen et al., 2018). Altering any block's data requires the regeneration of all subsequent blocks. The consensus mechanism is crucial in making modifications to multiple blocks prohibitively expensive, rendering such tampering virtually infeasible (Li, 2021). In proof-of-work blockchain networks, a user must control at least 51% of the computational power to alter data (Gervais et al., 2016). However, this is counterproductive for users with substantial computational power, thereby enhancing the reliability of blockchain data.

Smart contracts represent a digitally encoded set of commitments, encompassing agreements that delineate the conditions under which the participants can execute these commitments (Zou et al., 2021). These contracts facilitate transactions that are not only trusted but also traceable and irreversible, thereby eliminating the need for third-party intermediaries (Zheng et al., 2020; Liao et al., 2023).

As illustrated in Fig. 2, the *smart contract* is essentially a conditional code that operates on the blockchain. It comprises two main components: functions, which are the executable code units within the contract, and data, which represent the state of the *smart contract*. The contract terms, along with their corresponding trigger conditions and response rules, are predefined within the code. The *smart contract* is a collaborative agreement, mutually accepted and signed by all involved parties. It is submitted along with the user-initiated transaction, disseminated via the P2P network, and subsequently verified by miners before being stored in a specific block of the blockchain. Upon receiving the returned contract address and other relevant information, the user can invoke the contract by initiating a transaction. The system's preset incentive mechanism motivates miners to contribute their computational power towards transaction verification.

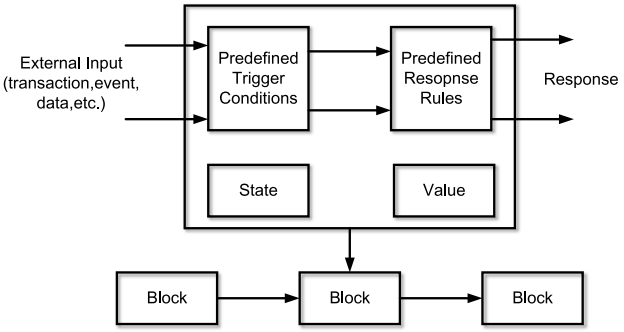


Fig. 2. The structure of a typical smart contract on Ethereum.

The trustworthiness and immutability of smart contracts primarily stem from two factors: First, once a smart contract is deployed to the blockchain network, its code and execution results are broadcast to the entire network, with all nodes storing and verifying this information (Luu et al., 2016). Additionally, the execution process of the smart contract is recorded on the blockchain and ordered sequentially with timestamps. This recording method ensures the chronological order and consistency of the information, and since the transaction records are publicly accessible, anyone can review and verify the execution of the smart contract, further enhancing its trustworthiness. Second, under consensus mechanisms such as Proof of Work (PoW), altering an already deployed smart contract would require an attacker to control more than 50% of the global network's computational power (Hu et al., 2020). Acquiring such a large amount of computational power would require substantial resources, making it extremely costly and virtually impossible to achieve. Therefore, once a smart contract is deployed on the blockchain, it is inherently immutable.

4. Approach

This section presents the technical details of *BlockSOP*.[1]

4.1. Main workflow of blocksop

Fig. 3 delineates the primary workflow of *BlockSOP*, which is composed of three principal modules: *Get Contribution Points Module*, *Contribution Points Usage Module*, and *Arbitration Module*. Our platform's goal is to ensure that all contributors to the software project, whether they are developers or investors, can transparently and fairly share the ownership and benefits of the project.

In the mechanism proposed in this paper, potential users can apply to join the project by investing or completing technical tasks. Upon successful admission, the *Get Contribution Points Module* is activated. This module requires users to contribute to the corresponding software project in order to earn *unexecuted contribution points*. These *contribution points* can be obtained through investment or technical contributions. Once users have successfully earned *unexecuted contribution points*, the *Use Contribution Points Module* is activated, allowing users to use these *unexecuted contribution points* to gain more code access permissions, which can then be converted into *contribution points*.

If malicious behavior occurs during the development process, the *project owner* can invoke the *Arbitration Module*. This module randomly selects users from a high-credit user database to form a *jury*, and the

[1] The BlockSOP is available at https://www.BlockSOP.com/

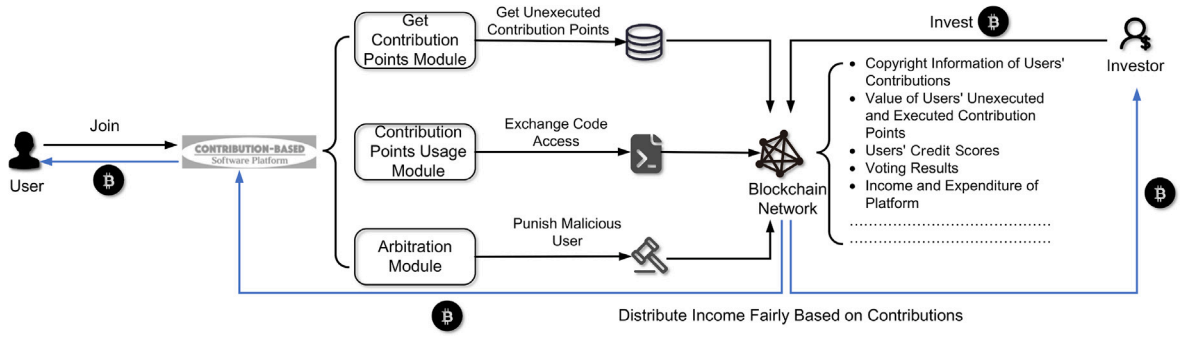


Fig. 3. Main workflow of the BlockSOP.

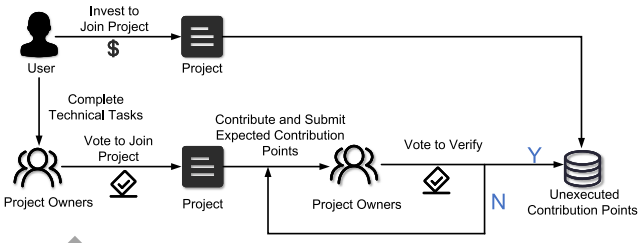


Fig. 4. The process of get contribution points module.

jury votes to determine the authenticity of the malicious behavior. If the malicious behavior is confirmed, the system will punish the malicious user. For example, deducting their corresponding *credit scores*.

Additionally, we record important information such as users' contribution copyrights, the amounts of *unexecuted and executed contribution points*, *credit scores*, voting results, platform income, and expenses on the blockchain network to ensure the fairness and transparency throughout the entire development and distribution process. The blockchain network will ultimately distribute income transparently and fairly based on users' *contribution points*.

4.2. Get contribution points module

Fig. 4 describes the process of getting *contribution points*.

Users can participate in a project in two ways. Firstly, by investing. Users can invest in our software projects by purchasing a trading contract. Once successfully purchased, the user will successfully join the software project and receive the corresponding *unexecuted contribution points*. The investment process helps users access more code information about our software projects because curious people from the open world can always pay for more code access, even if they cannot technically contribute. The money invested is also distributed according to the value of each contributor's contribution to the project. This process is also done through smart contracts to ensure fairness in the distribution process.

Secondly, by completing a technical task. When a user submits a completed technical task, a voting contract allows the *project owners* to review the technical task and evaluate whether the user can join the software project. Once a user successfully joins the project, he must make new contributions and submit the expected *contribution points*. If he receives votes from more than 80% of the *project owners*, he will successfully obtain the expected *contribution points*. But if he receives just over 60% of the votes from the *project owners*, the expected *contribution points* are invalid. Still, all project owners will provide their considered values for *contribution points*, and the final average will serve as the user's ultimate *contribution points* value. If he receives less than 60% of the votes, he will not receive any *contribution points*, and he will need to make new contributions.

The algorithm for the process of users getting *contribution points* is shown in Algorithm 1.

Algorithm 1 Get Contribution Points Algorithm

Input:
Number of Project Owners N;
Contribution Points Value V_i;
Expected Contribution Points Value V_e;
The Number of Assenting Votes from Project Owners $A; A \leq N$
Output: Final Contribution Points Value V_f
Submit Technical Tasks and Expected Contribution Points Value V_e
Project Owners Vote on V_e
if $A \geq 0.8N$ **then**
  $V_f = V_e$
else **if** $A \geq 0.6N$ **then**
  Set V_i by N Project Owners and Calculate Average Value
  $V_f = \frac{SUM(V_i)}{N}$
else
  $V_f = 0$
end if
Return V_f

In Fig. 4, the first voting instance, Vote to join project, aims to filter out suitable users for project participation. By allowing project members to vote, the system ensures that only qualified individuals are selected, thus guaranteeing that new members possess the necessary technical skills or ethical standards. This approach helps prevent low-quality contributions or malicious behavior from negatively impacting the overall quality of the project. The second voting instance, Vote to verify, is intended to assess the contribution points a user deserves. Relying solely on unilateral decisions for distributing contribution points could lead to unfair allocations or even manipulation. The voting mechanism decentralizes decision-making, ensuring that the allocation is collectively recognized, thereby reducing the risks of manipulation and abuse of the system.

The main distinction between these two methods of project participation lies in the speed at which users acquire *unexecuted contribution points*. Users who join the project through investment can immediately obtain *unexecuted contribution points* in proportion to their invested amount. Conversely, users who join by completing technical tasks must make new contributions, submit the tasks, and pass the vote of the current *project owners* to acquire *unexecuted contribution points*. It is

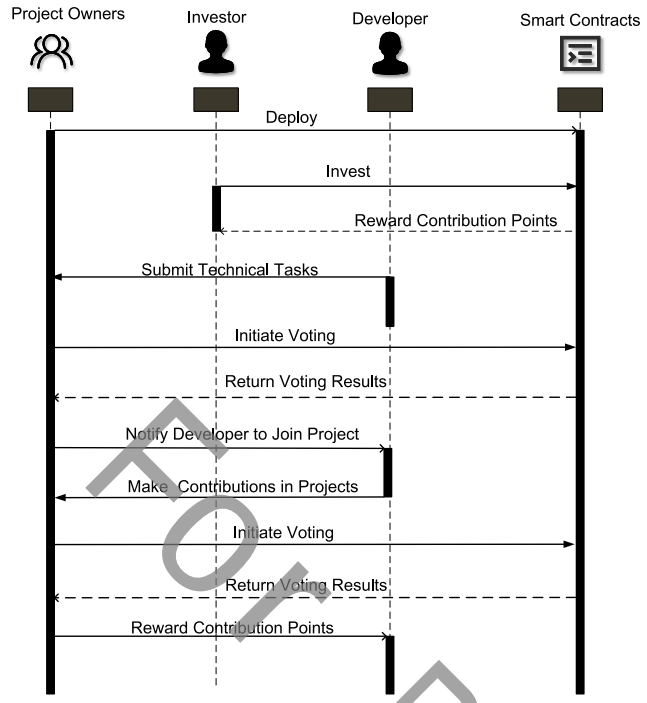


Fig. 5. Sequence diagram for developer and investor to obtain contribution points.

worth noting that different projects are independent of each other, and *contribution points* from one project cannot be used interchangeably with those of another project.

Fig. 5 shows the time sequence diagram for developer and investor to obtain *contribution points*. It is obvious that investment is a faster way to obtain *unexecuted contribution points*.

The time complexity of the Get Contribution Points Module is approximately $O(N)$, as it requires iterating over N project owners for voting.

4.3. Contribution points usage module

Fig. 6 illustrates the entire process of using *contribution points*. When a user aims to acquire additional code access rights, he needs to exchange their *contribution points* through smart contracts. The contract examines whether the user's *contribution points* are executed or unexecuted. If the *contribution points* are executed, they cannot be used to obtain more code access rights. However, if they are unexecuted, the points can be exchanged based on the predetermined exchange rate set within the software project. The corresponding *unexecuted contribution points* are converted into *executed contribution points* upon successful exchange. The *executed contribution points* cannot be further exchanged for code access rights. However, both unexecuted and executed *contribution points* grant contributors equal software profit-sharing rights. Software profit refers to the income distribution that occurs whenever a user invests in our software project. Smart contracts allocate the income based on the proportional value of *contribution points* held by each contributor in the project.

In the Contribution Point Usage Module, the redemption of contribution points is performed via a smart contract, with a time complexity of approximately $O(1)$.

4.4. Arbitration module

Fig. 7 describes the workflow of the *Arbitration Module*, which is a key component in ensuring that the entire development process is safe

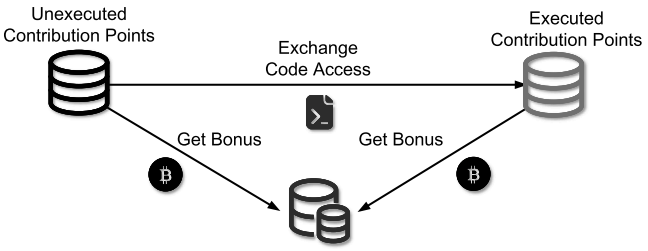


Fig. 6. Contribution points usage module.

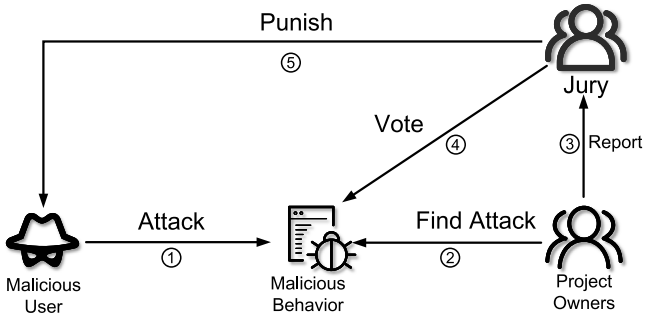


Fig. 7. Arbitrary module.

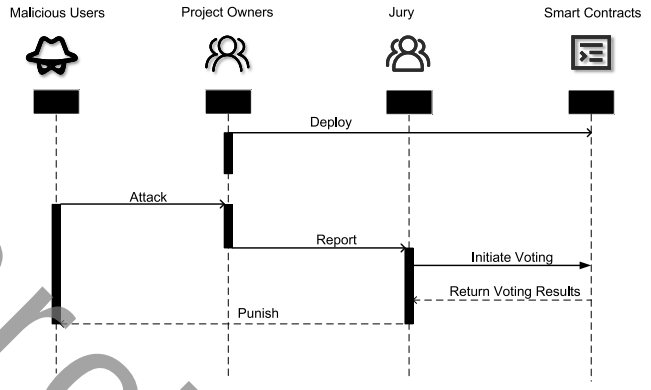


Fig. 8. Sequence diagram for arbitration.

and secure. This module is particularly important to reduce the risk of malicious behaviors.

When *project owners* identify potentially malicious behavior, they initiate the arbitration process. The *jury* is composed of K highly reputable users randomly selected from the M platform users. The *jury*'s task is to invoke the voting contract and vote on the authenticity and severity of the user's malicious behavior. If the user's malicious behavior is confirmed, the platform will impose punitive measures based on the severity of the malicious behavior. For example, deducting the user's *credit scores*, where the *credit scores* are directly proportional to the *contribution points* and earnings the user can obtain. Thus, a decrease in *credit scores* reduces the user's potential earnings. In severe cases, it may result in the user being blacklisted and barred from further project participation. Further details regarding the *credit scores* can be found in Section 5.2.

Fig. 8 describes the timing diagram of the arbitration process, which first requires the *project owners* to deploy a smart contract for arbitration voting. When a malicious user attacks, the *project owners* can initiate a report to the jury, invoke the voting function in the arbitration contract and eventually punish the malicious user according to the voting result.

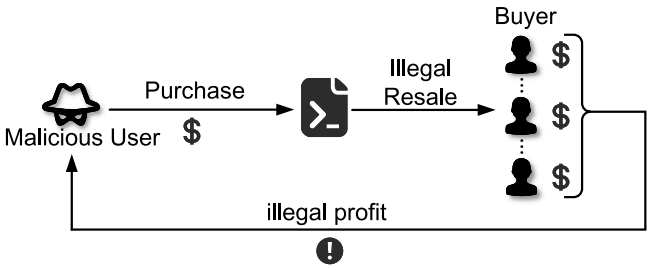


Fig. 9. The leakage behavior of malicious users.

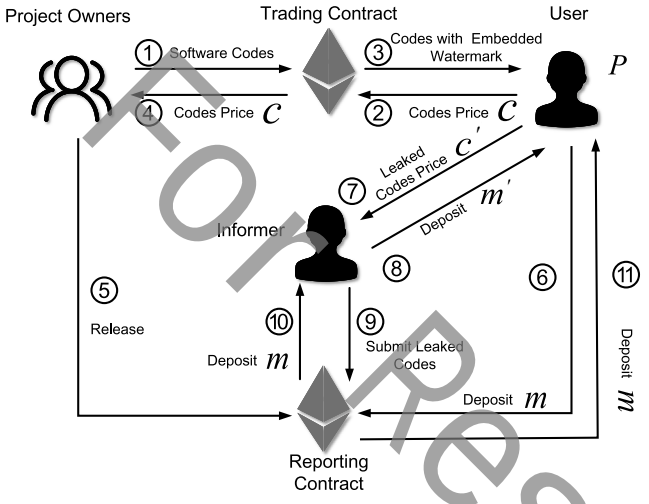


Fig. 10. The process of security mechanism.

In the Arbitration Module, *BlockSOP* first requires sorting the reputation of M users, with a time complexity of approximately $O(MlogM)$. Subsequently, K high-reputation users are randomly selected to vote, and the time complexity of the voting process is approximately $O(K)$. Therefore, the overall complexity of the arbitration module is approximately $O(MlogM + K)$.

5. Blockchain security mechanism

In this section, we delve into the potential issue of code leakage in software projects. Malicious users may invest in purchasing software codes from the platform and then illegally resell it for personal gain. Such code leakage can result in significant financial losses to our project and severely undermine the interests of platform users. The entire process of illegal resale is illustrated in Fig. 9.

To safeguard the interests of all platform users, we have implemented a robust security mechanism to prevent malicious users from leaking code information. This includes a deposit mechanism based on smart contracts and *code watermarking* technology, as well as a *credit score* system built on blockchain, which provides corresponding rewards and punishments to users.

Assuming that most malicious users are profit-driven, our goal is to maximize the cost of information leakage for malicious users while promptly rewarding informers. This creates a competitive profit landscape, compelling users to refrain from leaking code information. From a technical standpoint, the rewards and penalties of *BlockSOP* are immediately enforced.

The steps of the smart contract based deposit mechanism to resist code leakage are outlined below, with detailed interactions illustrated in Fig. 10.

Step 1: The *project owner* publishes the trading contract and stores the software codes to be sold in the trading contract.

Step 2: The user P who wants to purchase the software codes must transfer the amount of c to the contract account.

Step 3: The trading contract delivers the software codes with an embedded watermark to the user. Furthermore, each contributor will generate a bonus in the amount of rc, where r signifies P's contribution ratio to the overall contribution of the software.

Step 4: The trading contract transfers the amount c from the contract account to the *project owners*' accounts.

Step 5: At the same time as the trading contract is released, the *project owner* releases the reporting contract. All users who purchase the software codes must deposit m into the account of the reporting contract with a maintenance duration of t.

Step 6: The user deposits the deposit m into the account of the reporting contract.

Step 7: The leaker makes a code leakage transaction for the purchased software at the price of c'. It is worth mentioning that in this transaction, the leaker can sign a confidential contract with the purchaser to prevent himself from being reported and require the purchaser to pay a deposit of m' with a maintenance duration of t'.

Step 8: The informer transfers the deposit m' to the leaker and receives the leaked software codes.

Step 9: The informer submits the purchased leaked software codes to the reporting contract, thus reporting the code leakage.

Step 10: The watermark in the leaked software codes is extracted for verification, and if the code leakage is true, the leaker's deposit of m will be transferred to the informer. In addition, his *credit scores* will also be deducted.

Step 11: If the user does not leak any code information within the duration of t, then m will be refunded to the user in full.

5.1. Blockchain watermark traceability module

Code watermark (Li et al., 2023; Sun et al., 2023) is the practice of embedding specific markers or information within software or code, akin to digital watermarks, to enable tracking, identification, or protection of the code. This technique is often used to safeguard intellectual property, prevent piracy, track the use of source code, or authenticate the legitimacy of code (Miller et al., 2004; Kirchenbauer et al., 2023).

Code reordering facilitates the discreet insertion of watermarks, making them harder for leakers to detect while also enabling the tracking of the leaker. This paper implements *code reordering* by repeatedly altering the execution order of any two lines of code within a code segment S, ensuring the program executes correctly. As shown in Algorithm 2, the specific algorithm starts by detecting all possible pairs of line numbers within the given code segment that can be swapped, storing them in a collection L. Each pair of line numbers$\langle L_i, L_j\rangle$ in L must satisfy the following conditions: exchanging these two lines of code will not affect the correct execution of the program. Then, a subset L' is randomly and non-repetitively selected from L, and for each pair of line numbers $\langle L_i, L_j\rangle$in L', the corresponding code lines are swapped to generate a new code segment S'. Simultaneously, L' is saved as watermark information in the watermark database P on the server. Once code leakage occurs, the user corresponding to the watermark of the leaked code can be identified through the saved watermark information.

After generating the new code segment S' through *code reordering*, in the event of code leakage, the user responsible for code leakage can be identified through the saved watermark information library corresponding to the *code watermark*. Since the watermark information L' associated with each user is unique, it also identifies the user's identity. Algorithm 3 outlines the specific procedure. Initially, the algorithm retrieves all watermark information from the watermark library P. For each watermark information L', it sequentially extracts all pairs of line numbers $\langle L_i, L_j\rangle$ contained in L', swaps the code lines corresponding to the line number pairs, and converts the leaked code block M into M' accordingly. Then, the algorithm calculates the similarity between each transformed M' and the original code segment S, adding each

Algorithm 2 Code Reordering Watermark Algorithm

Input: Code Segment S
Output: S'
1: Detect All Interchangeable Code Pairs and Generate Set L=< L_1, L_2 >, ⋯ , < L_{n-1}, L_n >
2: Randomly and Non-repeatedly Select $L' \in L$
3: **for** $< L_i, L_{i+1} > \in L'$ **do**
4: $< L_i, L_{i+1} > \rightarrow < L_{i+1}, L_i >$
5: **end for**
6: Generate New Code Segment S'
7: Add L' to Set P
8: **Return** S'

similarity score to the collection N. After adding all similarities, it identifies the watermark information L' corresponding to the maximum value N_i in the collection N, thereby determining the user who leaked the code information.

Algorithm 3 Watermark Detecting Algorithm

Input:
1: Code Segment S
2: Leaked Code Segment M
3: P in Algorithm2
Output: Max Similarity
4: **for** $L' \in P$ **do**
5: **for** $< L_i, L_{i+1} > \in L'$ **do**
6: $< L_i, L_{i+1} > \rightarrow < L_{i+1}, L_i >$
7: Get One Copy of The Extracted Watermarking Code M'
8: **end for**
9: Calculate Similarity $M, S \rightarrow$ Set N
10: **end for**
11: **Return** Max of N_i
12: Get the Watermarked User Corresponding to N_i

The complexity of the watermarking algorithm primarily depends on the number of lines L and the complexity is approximately $O(L^2)$.

5.2. Credit module

The Credit Score (CS) directly indicates the quality of a user's development activities. This score is recorded via blockchain technology, ensuring transparency and fairness throughout the process. The *CS* is bifurcated into two components: the score accrued by users (positive *CS*) and the score deducted from users (negative *CS*). To provide a more precise measure of a user's credit, we aggregate the positive and negative *CS* across all projects. The *CS* is unique to each user, and the final *CS* is computed by adding the base *CS* to the positive *CS* and subtracting the negative *CS*.

The cumulative positive *CS* (P) and negative *CS* (N) are calculated as follows:

$$P = \sum_{j=1}^{n} \sum_{k=1}^{m} PC_{jk} \tag{1}$$

$$N = \sum_{j=1}^{n} \sum_{k=1}^{m} NC_{jk} \tag{2}$$

Here, we assume that there are n projects, and the maximum number of development activities a user can engage in across these n projects is m. PC_{jk} and NC_{jk} represent the *credit score* accrued and deducted, respectively, for the kth development activity in the jth project.

A user's initial *credit score* is set at 100. If it falls below 60, the user is barred from further project participation. P_{final} denotes the final *contribution points* the user can get, $P_{initial}$ represents the initial *contribution points* through project owners' voting, and P_{credit} signifies the user's current *CS*. The final *contribution points* is calculated as follows:

$$P_{final} = P_{initial} \cdot \frac{P_{credit}}{100} \left(P_{credit} \geq 60\right) \tag{3}$$

Eq. (3) shows that the *contribution points* the user finally receives are directly proportional to the *credit score*.

B_{final} represents the final reward that the user can obtain in the project, B_{sum} represents the total income of the project, and P_{sum} represents the total *contribution points* in the project. The final incentive allocation calculation is as follows:

$$B_{final} = B_{sum} \cdot \frac{P_{final}}{P_{sum}} \tag{4}$$

It is worth mentioning that *contribution points* and incentive allocations in different projects are independent of each other and not interchangeable.

The growth of a user's *CS* (S) is determined by two factors: the number of successful contributions (x) and the duration of continuous contributions (t), as shown below:

$$S = e^{-e^{-(t-365)}} + e^{-e^{-(x-200)}} (0 \leq t, 0 \leq x \leq 10^5) \tag{5}$$

We encourage users to contribute frequently and consistently, thereby enhancing the sustainability of our project. While it may be challenging for part-time developers to contribute consistently, they can still accrue *CS* through successful contributions. Users who maintain a high frequency of contributions and contribute over an extended period will experience a faster *CS* growth. Notably, the contribution count x is capped at 100,000, beyond which additional contributions will not increase the *CS*. However, the duration of continuous contribution t is not capped.

Comparing the deposit mechanism and the *CS* mechanism for copyright protection, the former is designed to counteract early-stage code leakage and provide monetary compensation to *project owners* in case of code leakage during the deposit period. At the same time, it imposes high penalties for leakers and prompt rewards for informers. Different from the deposit mechanism, the *CS* mechanism aims to reduce the likelihood of malicious behavior in the long run and encourage users to contribute consistently and frequently to enhance their *CS*. This fosters a transparent and secure environment for collaborative software development. Collectively, these two mechanisms complement each other, bolstering the security of our platform and minimizing the potential for malicious behavior.

In addition to this, in response to advanced fraud and denial-of-service (DoS) attacks in blockchain environments, *BlockSOP* also incorporates following strategies to enhance system robustness. First, *BlockSOP* adopts a separate architecture of mainchain and sidechain, where high-risk or high-load operations are offloaded to the sidechain while the mainchain is reserved for low-risk, lightweight tasks. This separation significantly reduces the mainchain's exposure to advanced fraud attacks. Second, by employing consensus mechanisms such as Proof of Stake (PoS), *BlockSOP* limits attackers' ability to create fake identities or nodes, thereby mitigating the risk of DoS attacks. Additionally, *BlockSOP* leverages the *credit score* to encourage honest behavior among participants. Malicious actions are penalized through deductions in reputation scores or contribution points, creating a robust deterrent against adversarial behaviors.

6. Analysis and evaluation

We implemented the smart contracts related to *BlockSOP* using Solidity 0.4.25 and tested them on the Ethereum test network. The experiments were conducted on a Lenovo R7000p laptop equipped with an AMD 3.00 GHz 8-Core CPU and 16 GB RAM.

Initially, we sought to assess the security of our platform, aiming to address the following key research question:

Table 2
Benefits and Costs Associated with the Roles of the Potential Pirate P and the Pirate Customers Depicted within the Game Tree Presented in Fig. 11.

Decision	(Response, Situation) Description	Benefits of P	Longterm Cost of P	Customer Profits
D_1	(R_1, s_1) No Piracy	nrc	-	–
D_2	(R_2, s_2) Piracy is reported during t	$nc' - m$	Further Lose Credits and Contributions	m
	(R_3, s_3) Piracy is reported after t	nc'	Further Lose Credits and Contributions	–
	(R_4, s_4) Piracy has never been reported	nc'	Further Lose Credits and Contributions	–
D_3	(R_2, s_5) Piracy is reported during t	$nc' + m' - m$	Further Lose Credits and Contributions	$m - m'$
	(R_5, s_6) Piracy is reported between t and t'	$nc' + m'$	Further Lose Credits and Contributions	$-m'$
	(R_6, s_7) Piracy is reported after t'	nc'	Further Lose Credits and Contributions	–
	(R_4, s_8) Piracy has never been reported	nc'	Further Lose Credits and Contributions	–

RQ1 How effective are the security mechanisms employed by *BlockSOP* in reducing the risk of software project information leakage?

Given the myriad attack scenarios, obtaining empirical answers to this research question about security proves challenging. As an alternative, we employed Game Theory to analyze the underlying mechanisms and draw conclusions accordingly. Furthermore, we conducted a series of experiments designed to answer additional pertinent research questions related to the effectiveness and efficiency of *BlockSOP*, as listed below:

RQ2 To what extent does *BlockSOP* enhance the motivation for software development?

RQ3 Does *BlockSOP* facilitate a more expedited completion in software development?

RQ4 What is the magnitude of the blockchain overhead associated with the *BlockSOP*?

6.1. RQ1: Analysis combining game theory

Game Theory is an interdisciplinary field, blending elements of mathematics and economics to systematically analyze and model strategic interactions between rational agents across various contexts (Wang et al., 2024; Ge et al., 2024; Liu et al., 2024). In this study, we employ *Game Theory* to examine the potential instances of software information leakage by users in *BlockSOP*.

We collectively refer to malicious users who profit from software code leakage as pirates. We assume that all users are rational, motivated by profit maximization, and make decisions independently of one another. By constructing a game-theoretic model of the software codes leakage process in *BlockSOP*, we aim to establish the following proposition: *For any user who purchases software codes in BlockSOP, the profits gained from software codes leakage or piracy do not exceed the profits derived from non-leakage, irrespective of whether the user has a confidentiality agreement with the individual to whom the information is disclosed.*

Our analysis commences with a user, denoted as P, who acquires software from the *BlockSOP* platform. The blockchain mechanism requires P to submit a deposit with a maintenance duration of t and an amount of m to deter piracy, alongside the cost of the licensed software, c. Furthermore, each authentic user will generate a bonus for P in the amount of rc, where r signifies P's contribution ratio to the overall contribution of the software. Concurrently, r dictates the proportion of software information accessible to P, which is also the portion of software information P may disclose. Piracy becomes viable only when r is not very small, indicating that the volume of information leaked by P is valuable for pirate purchasers.

Fig. 11 portrays a game tree that illustrates the decision-making process of a potential software pirate, referred to as P, when all conditions conducive to piracy are met. P confronts three distinct options: D_1, abstaining from both leaking and pirating software information; D_2, divulging software information without entering a confidential agreement with buyers of pirated products; and D_3, disclosing software information while signing a confidential agreement. The confidentiality agreement necessitates each pirate buyer to submit a deposit of m' to maintain the secrecy of P's piracy for a maintenance period t' that exceeds t, guaranteeing that P can reclaim his deposit m. Since the price of a licensed copy surpasses the price of a pirated copy, few pirate buyers are willing to provide a larger deposit than a licensed copy. Consequently, m' is set to be less than m.

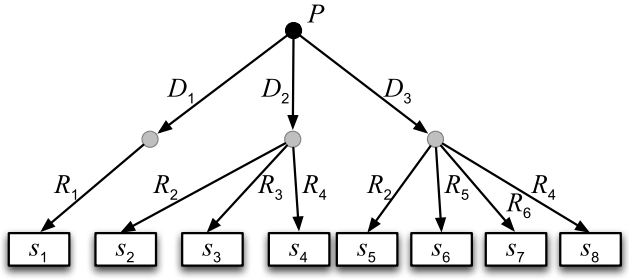


Fig. 11. The game tree depicting the decision-making process for a potential pirate P.

As depicted in Fig. 11, when P makes a decision D_1, D_2, or D_3, potential pirate buyers are able to react in six distinct ways ($R_1 - R_6$), culminating in eight unique scenarios ($s_1 - s_8$). Table 2 details the associated benefits and costs of P and potential pirate buyers in each response and situation. In this context, R_1 is the sole response for the decision when P selects D_1, which prevents the pirate and disclosure of software information. Consequently, potential buyers who adopt the R_1 response have no alternative but to purchase the legitimate software, yielding a revenue of $n \times rc$ for P, where n denotes the number of buyers, r indicates the proportion of P's contribution relative to the overall contribution of the software, and c represents the cost of the genuine software.

Upon choosing to pirate the software with decision D_2 without a confidentiality agreement, pirate customers can opt for responses R_2, R_3, or R_4, leading to scenarios s_2, s_3, or s_4, respectively. These customers will select the response that maximizes profits, which, as demonstrated in Table 2, is R_2. In this scenario, customers receive the deposit m as a reward for reporting piracy during the maintenance period t. Concurrently, P forfeits his deposit m and earns piracy revenue $n \times c'$ at the expense of losing the legitimate share $n \times rc$, where c' denotes the piracy price. Moreover, P's long-term relationship with *BlockSOP* suffers, leading to a decrease in his credit and contribution points. Regardless of whether the piracy is reported or not, *BlockSOP* can identify the leaked software version using watermarking techniques and ultimately impose penalties on P's credits and contributions.

Similarly, when P decides to pirate the software with decision D_2 while entering a confidentiality agreement, pirate customers have their options to respond with R_2, R_5, R_6 or R_4, resulting in scenarios s_5, s_6, s_7 or s_8, respectively. The maximum-profit scenario for a pirate customer remains R_2, involving reporting the piracy during the maintenance period t. In this scenario, the customer obtains the deposit m but loses m'. As $m' < m$, the customer is still profitable. Concurrently, P also

forfeits the deposit m but obtains the customer's deposit m' via the confidentiality agreement, earning piracy revenue $n \times c'$ at the cost of losing the legitimate share $n \times rc$. In the long run, P will still be detected by *BlockSOP* and lose his credits and contributions.

Based on the analysis in Table 2, for piracy on our platform to be profitable even in the short term, P must ensure that $nrc < nc' - m + m'$. However, this is challenging to satisfy under the constraints of piracy demand, as the pirate price c' must be significantly smaller than c, r cannot be too small for the leaked information be complete and valuable, and $m' < m$ since pirate buyers will not place a larger deposit than for a licensed copy. It is important to emphasize that P's long-term losses are even more substantial. A decline in credits and *contribution points* results in P losing considerable long-term revenue on the platform and precludes future piracy opportunities. Consequently, the optimal choice for P on our platform is to refrain from engaging in piracy.

Answer to RQ1: *BlockSOP* serves as a potent solution in mitigating the risks associated with software information leakage and unauthorized distribution by strategically integrating security, credibility, and contribution mechanisms.

6.2. RQ2: Motivation enhancement

To address Research Question 2 (RQ2), we conducted a comparative analysis of ask completion rates between the *BlockSOP* and BountySource platforms, both of which provide financial incentives for the successful completion of specified tasks. As Gitcoin's financial data is not accessible, we only compare its on-chain data later in Section 6.4.

For *BountySource*, we followed the methodology of Zhou et al. (2021) to collect 669 reports documenting the status of bounty tasks from January 1, 2019, to December 31, 2022. It is worth mentioning that after that, the BountySource platform was temporarily closed. A task was designated as *completed* if the bounty hunter successfully fulfilled the task and received the reward. Conversely, a task was labeled as *failed* if it was undertaken by a bounty hunter but remained unfinished for an extended period, surpassing the task's time limit. Finally, tasks that were never taken up by any bounty hunter before expiration were categorized as *underappreciated*.

To explore the characteristics of *BlockSOP* further, we invited software professionals to participate in project development on the *BlockSOP* platform by sending emails and inviting them to fill out an online survey upon completion of the development. In total, we sent out 1634 emails, invited 186 professional developers to work on *BlockSOP*, collected 482 task reports and 128 valid surveys[2] on *BlockSOP* from January 1, 2019, to December 31, 2022. A task was deemed *completed* if the *project owner* released a completed task and the developer received the corresponding contribution points. If a developer took on a task but failed to complete it, the task was marked as *failed*. Tasks that were not noticed by any developer before their expiration were considered *underappreciated*.

Developers were informed in advance about the purpose of our research and agreed to use the development results and surveys for research purposes. To expand the scope of the study, we invited participation from various companies globally. To recruit a sufficient number of developers from diverse backgrounds, we employed the following strategies to recruit developers:

[2] The survey is available at https://docs.google.com/forms/d/e/1FAIpQLSfa7h83sOoFsmU5Xyi2xxStHB6Y9Ai_yvkKdQlQGNw-aqjHWg/viewform

Table 3
Demographics of invited developers on BlockSOP.

Demographics	Statistics	%
Age		
<18	5	3.9%
18–24	12	9.4%
25–34	84	65.6%
35–44	24	18.8%
> 45	3	2.3%
Gender		
Female	24	18.8%
Male	100	78.1%
Prefer not to say	4	3.1%
Job Role		
Front-end development	30	23.4%
Back-end development	43	33.6%
Full-stack development	36	28.2%
Testing	16	12.5%
Other	3	2.3%
Professional Experience		
0-1 years	3	2.3%
1-3 years	19	14.8%
3-5 years	40	31.3%
5-10 years	53	41.4%
>= 10 years	13	10.2%
Experience in OSS Development		
Yes	110	85.9%
No	18	14.1%

- We contacted professionals from around the world and asked them to help disseminate our survey within their organizations. We sent emails to contacts at companies such as Google, Microsoft, Alibaba, Baidu, Tencent, ByteDance, Lenovo, and others, encouraging them to spread our survey among their colleagues. Through this strategy, we aimed to recruit developers from different organizations within the industry.
- Additionally, we also emailed contributors to 30 popular open-source repositories on GitHub, inviting them to participate in project development on the *BlockSOP* platform. Our goal was to recruit experienced contributors in open-source software development and professionals working in the industry.

The developers we invited to come from 7 different countries, as shown in Fig. 12. The top three countries where developers reside are China, the United States, and Russia.

As shown in Table 3, the developers we invited differ in age, gender, job roles, professional experience, and OSS development experience.

Since specific information about bounty hunters is not available on BountySource, we are unable to analyze their details. However, for better comparative analysis, in Table 4, we classified the types of issues/tasks on BountySource and *BlockSOP* to ensure that their distribution proportions across different types and difficulty are similar. We divided the types of issue reports into 6 categories, namely Feature, Improvement, Documentation, Bug, Design, and Code Review.

Fig. 13 presents a comparative analysis of task completion rates between the two platforms, *BlockSOP* and BountySource. The financial incentives provided by BountySource culminate in 20% of bounty tasks being successfully completed, while 67.3% are attempted but ultimately failed. Notably, 12.7% of bounty tasks remain underappreciated by developers until they expire. It is pertinent to highlight that over half of these underappreciated bounty tasks possess an expiration date exceeding one year, indicating a protracted period of inattention. In contrast, *BlockSOP* exhibits a 32.4% success rate in task completion, with 56.8% of tasks being unsuccessful and a mere 10.8% remaining underappreciated. This evidence underscores the enhanced efficacy of *BlockSOP*'s incentive structure in motivating developers compared to BountySource.

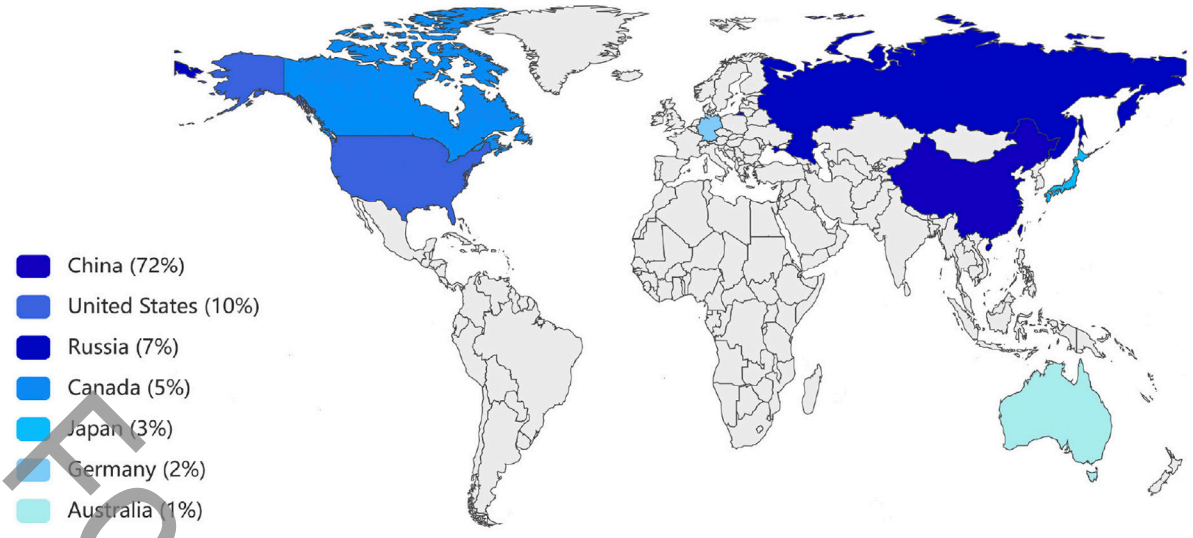


Fig. 12. Countries in which survey respondents reside. The darker the color is, the more respondents reside in that country.

Table 4
Types of Issues/Tasks.

Type	Difficulty	Description	BountySource(%)	BlockSOP(%)
Feature	General	Creating new features	42.4%	41.9%
Improvement	General	Improving existing features	16.8%	17.6%
Documentation	Easy	Creating documentation	7.4%	7.6%
Bug	General	Fixing bug	21.3%	20.9%
Design	Hard	Designing system	10.8%	10.3%
Code Review	Easy	Reviewing code	1.3%	1.7%

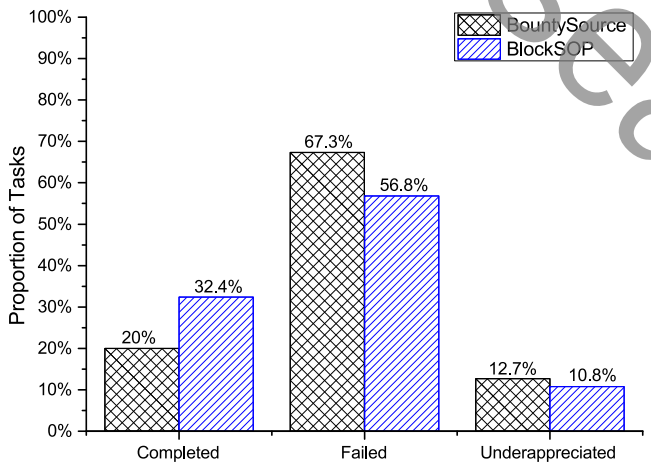


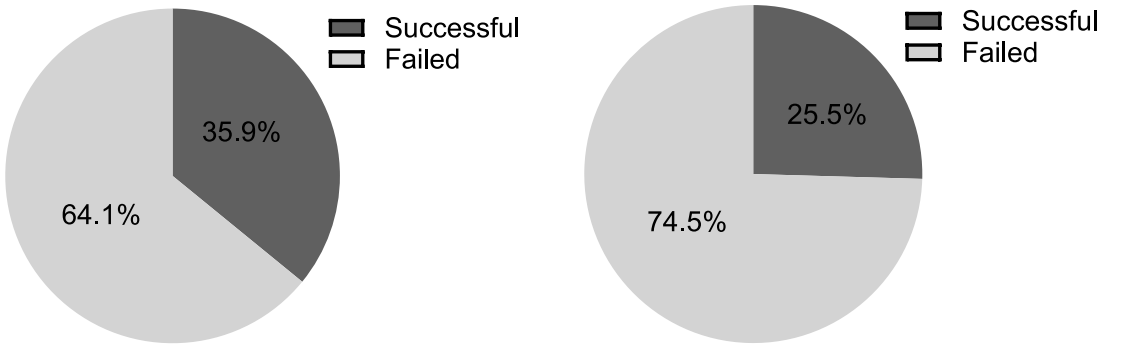
Fig. 13. Completeness status for tasks on BlockSOP and BountySource.

A thorough examination of the platform reveals several factors contributing to the superior motivational capacity of *BlockSOP*. Firstly, *BlockSOP* employs a contribution point-based mechanism that allocates revenue in accordance with each developer's input, a feature absent in BountySource. As opposed to BountySource, which solely rewards the successful bounty hunter, this equitable distribution mechanism fosters increased engagement from developers on *BlockSOP*, ultimately resulting in higher task completion rates. The disparity in completion rates between *BlockSOP* (32.4%) and BountySource (20%) serves as a testament to this effect.

Moreover, *BlockSOP* supplements financial incentives with technical incentives. For a subset of open-source developers who prioritize technical aspects over monetary rewards, the platform's progressively increasing access model provides further motivation. By granting developers access to more technical details and high-quality projects as they complete tasks, *BlockSOP* appeals to this group's intrinsic motivation. Such an approach is less effective on a bounty-driven platform like BountySource, where financial incentives remain the sole focus.

It is also worth mentioning that for bounty tasks valued under $100 on BountySource, no funds are returned to the task originator, even if the task is incomplete or expired. As a result, BountySource sponsors are more thoughtful and cautious in posting tasks compared to those on *BlockSOP*. Despite this consideration, BountySource still exhibits a higher rate of failed tasks, which can be attributed to its inferior capacity for fostering multi-developer collaboration relative to *BlockSOP*. It is essential to recognize that numerous software tasks are complex, necessitating the collaboration of multiple developers who may not have prior familiarity with one another in order to effectively contribute their expertise. The single bounty hunter reward mechanism employed by BountySource is ill-suited for such situations. In contrast, the blockchain-based management system employed by *BlockSOP* capitalizes on the benefits of multi-developer open collaboration, effectively addressing this challenge and significantly reducing the task failure rate.

Additionally, we conducted a comparative analysis of the proportion of projects receiving sponsorship or investment on OpenCollective and *BlockSOP* in Fig. 14. Using the OpenCollective API, we collected donation data from 838 collectives on the OpenCollective platform from January 1, 2019, to December 31, 2022. We define projects that successfully received donations on OpenCollective as *successful* and those that did not as *failed*. Similarly, we define projects on *BlockSOP* that successfully received investment as *successful* and those that did not as *failed*. Our analysis reveal that the proportion of projects receiving investment on *BlockSOP* (35.9%) is slightly higher than that on OpenCollective (25.5%). This difference may be attributed to *BlockSOP*'s investment incentive mechanism, which allows investors to directly benefit from their investments. Additionally, its blockchain-based



(a) The investment ratio of projects in BlockSOP (b) The donation ratio of projects in OpenCollective

Fig. 14. Comparison of sponsorship or investment ratio between OpenCollective and BlockSOP.

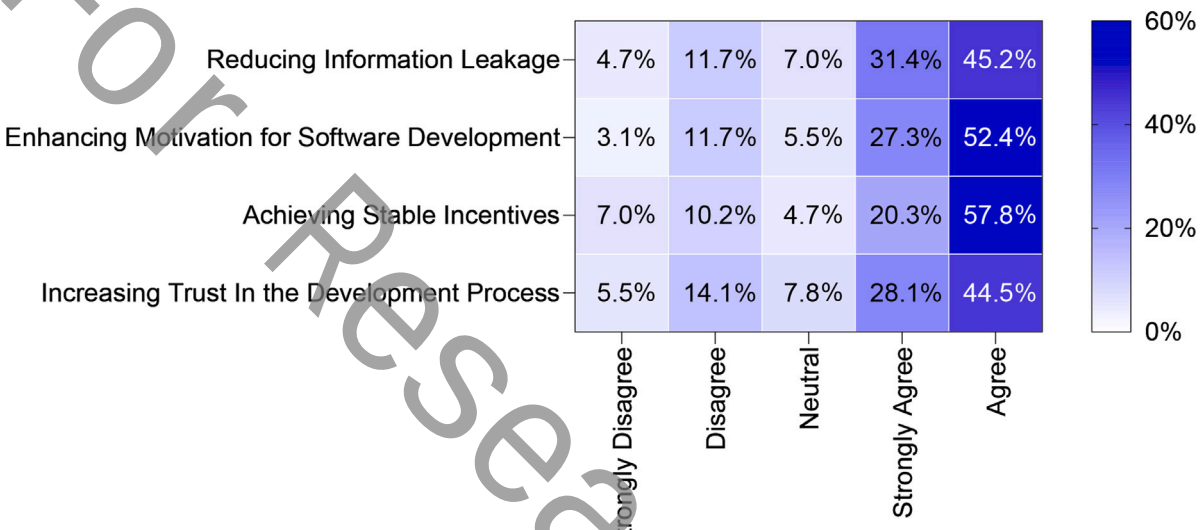


Fig. 15. Proportion of survey results.

decentralized design provides high transparency and traceability, effectively reducing investment risks and enhancing the credibility of projects, thus increasing investor trust in the platform. In contrast, OpenCollective lacks a similar financial return mechanism, with donors primarily motivated by altruistic purposes, making it less attractive to institutional investors. Furthermore, OpenCollective faces transparency issues regarding the use of funds, particularly for complex or long-term projects, which may undermine donor trust.

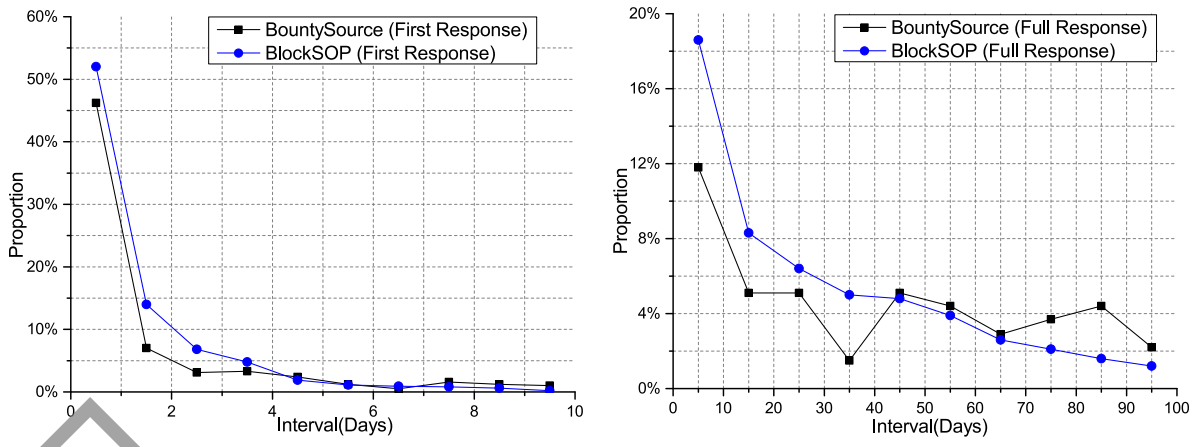
According to the results of 128 valid surveys[3] in Fig. 15, 76.6% of participants strongly agree or agree that *BlockSOP* can reduce the risk of software information leakage; 79.7% of participants strongly agree or agree that *BlockSOP* can enhance the motivation for software development; 78.1% of participants strongly agree or agree that stable incentives can be achieved on *BlockSOP*; 72.6% of participants strongly agree or agree that *BlockSOP* increases trust on development Process. The survey results of *BlockSOP* users demonstrate the effectiveness of *BlockSOP* in reducing the risk of software information leakage, achieving stable incentives, and increasing trust in the development process.

Answer to RQ2: *BlockSOP* integrates a multi-developer open collaboration mechanism and a multi-tiered incentive system, significantly boosting the enthusiasm for software development and providing developers with stable incentives. At the same time, *BlockSOP*'s security mechanisms also enhance trust during the development process.

6.3. RQ3: Response speed of development

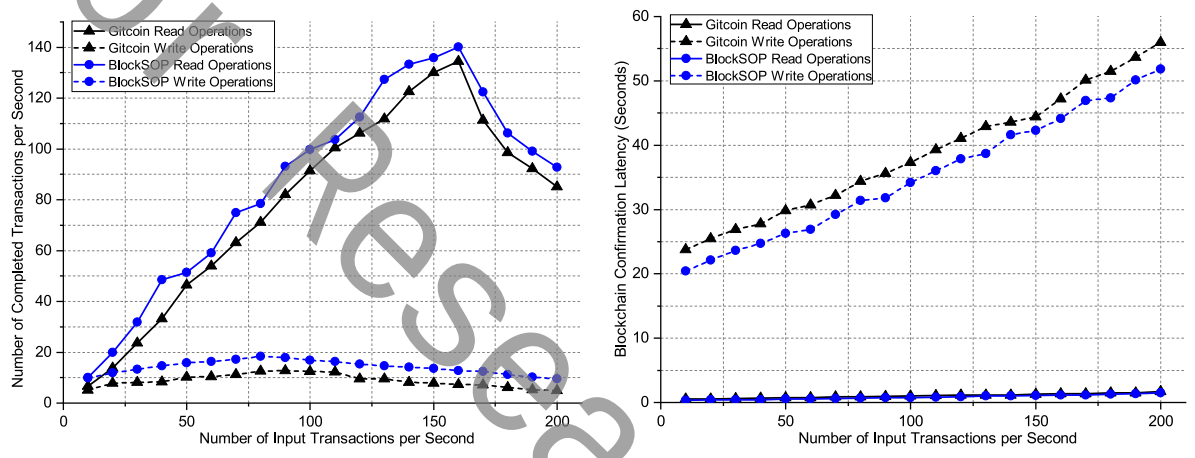
We compare the entire interval between a sponsor posting an issue on BountySource, proposing a bounty, and a user finally solving the issue and successfully receiving a bounty. The entire interval between the issue is published on *BlockSOP*, and the user successfully contributes to obtaining contribution points. From Fig. 16, We can see that in terms of first response time, the first response time of both BountySource(77.3%) and *BlockSOP*(81.5%) is concentrated in the 0–10 interval, but the percentage of *BlockSOP* in this interval is 4.2% higher than that of BountySource, while in terms of full response time, *BlockSOP*'s full response time(34.6%) is mainly concentrated in the 0–10 interval, while the whole response time of Bountysource(26.1%) is mainly in the range of 0–10, and the full response time is significantly longer than that of *BlockSOP*. There are two main reasons for this difference. Firstly, sponsors do not add bounties immediately after raising a problem on BountySource, and they may wait a while to see if the issue is resolved without a bounty. After an interval of waiting, if their problem is not solved, sponsors will start to offer bounties, and this waiting process will consume more time. At the

[3] The survey results are available at https://github.com/Shuoxiaozhang/BlockSOP



(a) First Response Compare (b) Full Response Compare

Fig. 16. Comparison of BountySource and BlockSOP first response time and full response time.



(a) Blockchain Throughput (b) Blockchain Latency

Fig. 17. Analysis of Blockchain performance for BlockSOP and GitCoin platform contracts under workload conditions ranging from 1 to 200 input transactions per second.

same time, *BlockSOP* does not have such a waiting process, and users will solve the issue without BountySource's waiting time. Second, there are some difficult issues on BountySource with large bounty amounts. Since BountySource only supports one person in getting the bounty, it takes a long time for a single bounty hunter to solve difficult issues. In contrast, the *BlockSOP* mechanism can solve such difficult issues through cooperative development by multiple users.

Answer to RQ3: Compared to BountySource, *BlockSOP*'s first issue response time and full response time are shorter, increasing developers' motivation and shortening issue resolution cycles.

6.4. RQ4: Blockchain overhead

Our *BlockSOP* platform is designed utilizing blockchain-based techniques to ensure transparency and trustworthiness. The overhead introduced by smart contracts on the blockchain is a crucial factor that inevitably constrains the platform's scalability. As a result, we have meticulously implemented our platform using Ethereum, a mainstream blockchain technology that boasts a capacity of approximately 40 transactions per second (tps) and a block time of around 15 s. In comparison to other prominent blockchain technologies, such as Bitcoin, which just supports around 7 transactions per second (tps) and features a block time of approximately 10 min, Ethereum demonstrates distinct advantages.

In order to rigorously evaluate the effectiveness of our proposed techniques, we utilized Hyperledger Caliper (Choi and Hong, 2021), a widely-recognized, industrial-grade performance assessment tool specifically designed for blockchain systems. We conducted a comparative analysis of the obtained results with blockchain data sourced from Gitcoin[4], a trailblazing platform that leverages blockchain technology to facilitate open-source software development (Patrickson, 2021), and the version of Gitcoin we use is 1.0.

Our investigation encompassed an examination of the throughput and latency associated with various blockchain operations. As illustrated in Fig. 17, these operations included both read operations, such as data queries and write operations, such as data updates. We submitted transactions to the smart contracts of both platforms at rates ranging from 10 to 200 transactions per second. Subsequently, we gathered data on the latency and the number of transactions completed per second using the Hyperledger Caliper tool to measure the performance of the two systems.

In Fig. 17, it can be observed that the performance of blockchain operations on *BlockSOP* marginally surpasses that of Gitcoin. Under the same environmental configuration and input parameters, *BlockSOP* executes, on average, 8.21 additional read transactions and 5.27 additional write transactions per second compared to Gitcoin. Furthermore,

[4] Gitcoin blockchain contracts are collected from https://github.com/gitcoinco/smart_contracts

Table 5
Performance analysis of BlockSOP under high workloads.

Workloads	Read throughput(TPS)	Write throughput(TPS)	Read latency(ms)	Write latency(ms)
500	487.23	493.68	38.12	48.56
600	573.45	578.92	43.23	53.78
700	660.89	668.27	49.56	58.91
800	738.12	746.84	54.68	66.79
900	818.64	826.09	59.12	77.23
1000	896.47	905.73	64.45	88.34

Table 6
Blockchain cost of transactions on BlockSOP.

Transaction	Gas	ETH	USDollar
Get contribution points	1 731 077	0.021	61.43
Contribution points usage	421 985	0.005	14.98
Arbitration	786 942	0.009	27.93
Software trading	639 083	0.008	22.68
Fraud reporting	823 471	0.010	29.22

Table 7
User experience of working on BlockSOP.

Dimension	Excellent	Good	Average	Below average	Poor
Interactivity	22.2%	33.3%	27.8%	11.1%	5.6%
Usability	33.3%	27.8%	22.2%	11.1%	5.6%
Functionality Adaptation	27.8%	22.2%	33.3%	5.6%	11.1%
Collaboration Efficiency	38.9%	33.3%	11.1%	11.1%	5.6%

Table 8
Cost comparison of Ethereum, Polygon, and Solana.

Metric(Average)	Ethereum	Polygon	Solana
Transaction Fee	$20	$0.001	$0.0001
Development Cost	Moderate	Low	High
Operating Cost	Moderate	Low	High
Safety	High	Moderate	Low

the blockchain confirmation latency on *BlockSOP* is 0.21 s shorter for read transactions and 3.33 s shorter for write transactions. This improved performance can be attributed to the allocation processes in *BlockSOP*, which utilize the contribution point mechanism. This mechanism simplifies the corresponding processes, thereby allowing for slightly superior optimization.

In addition to the aforementioned observations, several other noteworthy points emerge from the analysis. Firstly, the blockchain throughput gradually reaches a peak at approximately 160 input transactions per second for read operations and 90 input transactions per second for write operations. This observation implies that blockchain-based applications, including *BlockSOP* and Gitcoin, may encounter congestion if the workload becomes overwhelming, thereby limiting their scalability. To address this issue, our platform employs a buffering mechanism for transactions in cases where a significant number of users simultaneously submit operations. Furthermore, we plan to explore advanced blockchain techniques to accommodate large-scale transactions. Secondly, it is worth noting that the blockchain latency exhibits a linear relationship with the workload. This trend is considered acceptable in the long term. Given that the write latency is moderately elevated at around tens of seconds, we strive to optimize it by consolidating more write operations within a limited number of blocks.

BlockSOP is designed to balance security and load efficiency. The security of the Ethereum main chain is relatively higher than that of side chains, making it more suitable for performing low-load operations. Under high-load conditions, its operations place a significant burden on the Ethereum main chain, prompting migration to the Ethereum sidechain, Polygon, for handling high-load tasks. The data in Table 5 show that as the load increases, the read throughput and write throughput of the *BlockSOP* sidechain rises rapidly while read latency and write latency gradually increases. Write operations experience slightly higher latency than read operations. This reflects that, under high-load conditions, the sidechain's performance is somewhat impacted by resource bottlenecks, network congestion, and other factors. However, despite the slight increase in latency, the sidechain maintains a high read throughput and write throughput, and the system is still able to handle high concurrency, ensuring the normal operation of *BlockSOP* under high workloads.

In addition, we have assessed the gas cost of operations on the *BlockSOP* platform. Table 6 presents the average Ethereum gas consumption requisite for the execution of various types of transactions. We have also calculated the corresponding monetary value in US Dollars, considering the current market price of Ethereum, which stands at 2957.22 US Dollars per ether and the gas price is 12Gwei in April 2024. We can observe that the financial cost of using smart contracts on the Ethereum network is relatively low. The cost is generally associated with the computational and storage complexity of the functions. For instance, the *Get Contribution Points* function, due to its relatively complex workflow, consumes more gas compared to other functions. In addition, the volatility of Ethereum prices and the congestion level of the network also have an impact on the deployment cost of smart contracts. Therefore, our data is only specific to the current period's Ethereum prices and network conditions.

We also conduct a comparative cost analysis of Ethereum, Polygon, and Solana to assist projects with different budgets in selecting the most suitable solution. From the comparison data in Table 8 we observe that although Ethereum has a higher average transaction cost compared to Polygon and Solana, it offers the highest level of security, making it suitable for projects with high security requirements and sufficient budgets. For projects with moderate security needs and tighter budgets, Polygon is the ideal choice, as it provides lower transaction, development, and operating costs while maintaining moderate security. Although Solana has extremely low transaction costs, its higher development and operating costs, coupled with lower security, make it more suitable for smaller projects with less stringent security needs. Given that *BlockSOP* has high security requirements, and Ethereum, as a mature and widely used blockchain platform, can offer superior security to effectively mitigate potential attacks and vulnerabilities, it is the preferred choice for the platform.

Answer to RQ4: Compared to the state-of-the-art platform Gitcoin, the optimized *contribution points* mechanism of *BlockSOP* demonstrates better performance in terms of throughput and latency while maintaining reasonable costs.

6.5. User experience and feedback

In this section, we conduct a qualitative study on the user experience and feedback of *BlockSOP*. We interviewed 18 experienced users of the *BlockSOP* platform to gather insights into their work experiences and collect their feedback on the system. Based on the characteristics of the *BlockSOP* platform and the requirements of collaborative development, we selected the following key metrics to evaluate user experience:

Interactivity: Measures the overall user experience in interacting with the system on the platform.

Table 9
Feedback of BlockSOP from Users.

Weakness	Percentage	Strength	Percentage
1: Lack of help and guidance	22.2%	1: Fair process for allocating contribution points	27.8%
2: Need optimize user interface	16.6%	2: Reduce malicious behaviors	22.2%
3: Lack of expanded integration functions	27.8%	3: Effective copyright protection mechanism	16.6%
4: Need Improve platform performance	22.2%	4: Improve transparency of software projects	11.2%
5: Need Enhance security protection	5.6%	5: Improve efficiency of collaborative development	16.6%
6: Other	5.6%	6: Other	5.6%

Usability: Assesses the intuitiveness of the platform interface and the convenience of its operations.

Functionality Adaptation: Examines the platform's ability to meet diverse user needs.

Collaboration Efficiency: Evaluates the platform's capability to enhance efficiency in multi-user collaborative development.

Table 7 presents user evaluation data on the *BlockSOP* platform's user experience, revealing the following patterns: collaboration efficiency received the highest ratings, with 38.9% of users rating it as excellent and 33.3% as good, and only a small proportion rating it as average or below. This highlights the platform's outstanding performance in supporting multi-user collaborative development, demonstrating a clear advantage. Usability also scored relatively high, with 33.3% of users rating it as excellent and 27.8% as good, though 22.2% rated it as average, indicating that while the platform's interface and operational convenience are adequate, there is room for improvement. The ratings for interactivity and functionality adaptation were more dispersed, with 22.2% and 27.8% of users rating them as excellent, respectively, but a significant proportion rated them as average (27.8% and 33.3%). Moreover, 16.7% of users rated these aspects as below average or poor, suggesting that while the platform generally meets user needs for interactivity and functionality, further optimization is required. Functionality adaptation received the highest proportion of poor ratings (11.1%), indicating that the demand for more diverse functionality has not been fully met, and the platform needs to expand its feature set in the future.

Table 9 summarizes user feedback on the strengths and weaknesses of *BlockSOP*. Among the strengths, the most highly recognized feature is the fair process for allocating contribution points (27.8%), highlighting the platform's outstanding performance in incentive mechanisms and fairness. This is followed by its ability to reduce malicious behaviors (22.2%) and its effective copyright protection mechanism (16.6%), demonstrating the platform's strong competitiveness in security and contribution copyright protection. Additionally, improving the efficiency of collaborative development (16.6%) and improving transparency of software projects (11.2%) have gained user recognition, reflecting the platform's positive impact on collaborative development and transparency. On the weaknesses side, the most significant issue identified by users is the lack of expanded integration functions (27.8%), which reveals the platform's limitations in addressing diverse needs. Lack of help and guidance (22.2%) and the need to improve platform performance (22.2%) further indicate user demands for enhanced technical support and platform optimization. Additionally, the feedback proportions for user interface optimization (16.6%) and enhanced security protection (5.6%) suggest there is still room for improvement in user experience and security features.

In conclusion, user feedback indicates that *BlockSOP* demonstrates significant strengths in incentive mechanisms and security. However, further improvements are necessary in functional extensibility, user support, and platform optimization to better meet user demands and enhance the overall user experience.

6.6. Threats to validity

Despite the numerous significant advantages of smart contracts, they are still vulnerable to risks. The execution of a smart contract strictly depends on its code logic, and if there are flaws or design defects in the contract code, they could be exploited by attackers, potentially resulting in financial loss or unfair outcomes. For example, the infamous DAO Attack occurred due to a vulnerability in the contract code that was maliciously exploited by hackers.

The *BlockSOP* ensures data authenticity and integrity through security mechanisms such as smart contracts and code watermarking. However, scalability challenges may arise under high-load conditions. Smart contracts automatically execute protocols to ensure transparency and immutability, but scalability bottlenecks can occur during large-scale deployments. As the number of network nodes and transaction frequency increases, the execution time and resource consumption of smart contracts grow rapidly, resulting in higher Gas fees and longer transaction confirmation times, which limit system scalability. This issue is particularly evident in malicious behavior detection and voting processes, where smart contracts frequently perform high-load computations, placing a significant strain on the blockchain network. Similarly, code watermarking technology faces related challenges. As the project scales, the computational and storage burden for watermark generation and verification increases, especially when large-scale code updates are required, placing a substantial load on the platform.

In the game-theoretic proof, the object of our proof is the users who purchase software codes. The malicious behavior is mainly about code information leakage because these investors often have access to a large amount of software project information, so they need to be focused on prevention; for other users who do not make software investment purchases, they do not trigger the transaction contract and deposit contract, so they are not in the scope of our proof.

In addition, in comparing the response interval of issue solving, we consider all the cases where the user successfully gets the bounty in BountySource and do not consider the successful problem-solving reports in which the bounty is not received in the end.

Finally, in the test performance, the Ethereum test network may have network fluctuations, which will have an impact on the test results. For this reason, we have conducted multiple sets of tests and have taken the average value as the result.

7. Related work

As a typical virtual group development community, open-source software projects come from different countries and backgrounds. Developers collaborate with other developers by joining the community. Prominent open source communities include GitHub and SourceForge (Joo et al., 2012; Tamburri et al., 2020; Robinson and Vlas, 2015). Unlike traditional software projects, the development of open-source software projects has many advantages, such as low barriers and freedom of participation. However, there are undeniably some drawbacks and shortcomings regarding quality and sustainability (Alami, 2020; Yin et al., 2022; Gamalielsson and Lundell, 2012; Yin et al., 2021). The most significant one is the need for sustainability mechanisms. For example, prolonged interruptions or withdrawals of core developers leading to project suspension (Foucault et al., 2015; Zanetti, 2012; Miller et al., 2019) or the lack of effective incentives leading to unsustainable projects prevent open source projects from forming a virtuous cycle of effective feedback and continuous operation (Xu and Wan, 2008). The key to the sustainable development of open-source projects is to have a considerable number of active submitters

and users (Iaffaldano et al., 2019a) and to have a reliable management mechanism to balance the rights, responsibilities, and interests of multiple participants (Silva et al., 2017; Valiev et al., 2018b).

Donation and crowdfunding models are becoming increasingly popular in open source fields (Homscheid and Schaarschmidt, 2016). In May 2019, GitHub launched GitHub Sponsors (Zhang et al., 2023), a service that allows open-source software developers to accept donations directly from other GitHub users. While most open-source software donation services in the past have been project-specific, GitHub Sponsors is unique in allowing users to donate to individual open-source software developers. Even though the program has been live for over three years, some developers still complain that they need to get the sponsorship they expected. Open Collective is a transparent community management platform. Open Collective (Panigrahi et al., 2022) creates contribution levels that offer different benefits based on how much is donated and how consistently it is donated. It can set budget goals, manage community expenses online, provide open and transparent accounts, and other features.

Compared to traditional blockchain-based software management technologies, *BlockSOP* has pioneered a new direction in guiding contributions and ensuring fair incentive distribution. Traditional blockchain-based software management platforms, like Gitcoin, mainly record project information such as financial transactions and personnel details. In contrast, *BlockSOP* not only includes these functionalities but also defines clear criteria for earning contribution points through blockchain and smart contract mechanisms, encouraging user participation. The incentive distribution process is automated and transparent, eliminating intermediaries and ensuring that rewards are allocated fairly and promptly. Additionally, *BlockSOP* allows for flexible platform expansion and project customization, enabling the system to respond quickly to changes in the market and technological environment, thus meeting the personalized needs of developers.

8. Conclusion and future work

This paper presents a trustworthy software management platform based on blockchain (*BlockSOP*) to enhance user motivation in open collaborative development. We innovatively introduce the concept of shares into software collaboration, distributing project earnings fairly based on users' contribution percentages while ensuring users' due benefits through blockchain. To address potential malicious behavior and code leakage during the development process, we propose a security mechanism based on deposit and credit scores. Using game theory, we simulate user profits under different decision scenarios and conduct user surveys to demonstrate the reliability of the security mechanism and the trustworthiness of the entire development process.

Furthermore, to examine *BlockSOP*'s performance in enhancing user development motivation and response speed, we compared data from the BountySource platform with data from the *BlockSOP* platform. We found that the task completion rate on the *BlockSOP* platform was 12.5% higher than on BountySource. Additionally, the first response time and complete response time in short intervals were lower by 4.2% and 8.5%, respectively, compared to BountySource. This indicates that *BlockSOP* significantly improves user development motivation and response speed compared to BountySource.

Finally, we tested the performance and overhead of *BlockSOP* and compared it with the Gitcoin platform. The results indicate that *BlockSOP* outperforms Gitcoin slightly in terms of throughput and latency, and the overhead is also within a reasonable range.

Currently, *BlockSOP* simplifies user interaction by shielding most of the system's complex mechanisms through an intuitive and straightforward interface, allowing users to focus on core functions and operations without needing to understand the technical implementation in detail. For instance, the contribution evaluation and voting processes are designed with user-friendly interfaces where users can complete tasks with simple clicks and selections. Additionally, we provide guidelines for using *BlockSOP* to help users quickly familiarize themselves with the platform's workflow and highlight key considerations during usage. The platform also offers detailed technical documentation, example code, and tutorials to assist developers in swiftly mastering the workflow and encouraging active participation. Given that operations such as contribution point review submissions and reporting malicious behavior impose a significant load on the platform, the system provides reminders to users during these actions, discouraging frequent submissions in a short period to prevent excessive resource consumption. In the future, we plan to further optimize *BlockSOP*'s user interface and architecture, lowering the technical barrier for average developers and enhancing the overall user experience.

We also plan to extend *BlockSOP* to the field of intellectual property, where its adaptability is reflected in its blockchain-based copyright protection mechanism, effectively safeguarding creators' intellectual property. The platform can record and verify the copyright of creators' original works and automatically execute the distribution and transaction of intellectual property through smart contracts. For example, creators can register their works on the platform and set corresponding authorization terms through smart contracts, ensuring that their rights are protected during use or transfer. This decentralized management approach not only enhances the efficiency of intellectual property protection but also reduces the costs and risks associated with traditional intermediary institutions.

In addition, we will continue to optimize the performance of smart contracts and blockchain platforms to meet the demands of large-scale transaction scenarios.

CRediT authorship contribution statement

Shuoxiao Zhang: Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Enyi Tang:** Writing – review & editing, Supervision, Software, Methodology, Investigation, Formal analysis, Conceptualization. **Haoliang Cheng:** Writing – review & editing, Validation, Software, Resources, Data curation. **Xinyu Gao:** Supervision, Software, Resources, Project administration. **An Guo:** Supervision, Methodology, Investigation. **Jianhua Zhao:** Supervision, Software, Resources, Project administration, Methodology. **Xin Chen:** Supervision, Software, Resources, Conceptualization. **Linzhang Wang:** Supervision, Software, Investigation, Conceptualization. **Na Meng:** Supervision, Software, Methodology, Conceptualization. **Xuandong Li:** Supervision, Software, Resources, Project administration, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors would like to thank the anonymous reviewers for insightful comments. This research is supported by National Natural Science Foundation of China (Grant No. 62172210 and 62172211).

Data availability

Data will be made available on request.

References

Alami, A., 2020. The sustainability of quality in free and open source software. In: Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Companion Proceedings. ICSE '20, Association for Computing Machinery, New York, NY, USA, pp. 222–225.

Alzoubi, Y.I., Al-Ahmad, A., Kahtan, H., 2022. Blockchain technology as a fog computing security and privacy solution: An overview. Comput. Commun. 182, 129–152.

Assavakamhaenghan, N., Tanaphantaruk, W., Suwanworaboon, P., Choetkiertikul, M., Tuarob, S., 2022. Quantifying effectiveness of team recommendation for collaborative software development. Autom. Softw. Eng. 29 (2), 51.

Barcomb, A., Kaufmann, A., Riehle, D., Stol, K.-J., Fitzgerald, B., 2020. Uncovering the periphery: A qualitative survey of episodic volunteering in free/libre and open source software communities. IEEE Trans. Softw. Eng. 46 (9), 962–980.

Barcomb, A., Stol, K.-J., Fitzgerald, B., Riehle, D., 2022. Managing episodic volunteers in free/libre/open source software communities. IEEE Trans. Softw. Eng. 48 (1), 260–277.

Bonaccorsi, A., Rossi, C., 2003. Why open source software can succeed. Res. Policy 32 (7), 1243–1258. Open source software development.

Butler, S., Gamalielsson, J., Lundell, B., Brax, C., Sjöberg, J., Mattsson, A., Gustavsson, T., Feist, J., Lönroth, E., 2021. On company contributions to community open source software projects. IEEE Trans. Softw. Eng. 47 (7), 1381–1401.

Casadesus-Masanell, R., Llanes, G., 2015. Investment incentives in open-source and proprietary two-sided platforms. J. Econ. Manag. Strat. 24 (2), 306–324.

Choetkiertikul, M., Puengmongkolchaikit, A., Chandra, P., Ragkhitwetsagul, C., Maipradit, R., Hata, H., Sunetnanta, T., Matsumoto, K., 2023. Studying the association between Gitcoin's issues and resolving outcomes. J. Syst. Softw. 206, 111835.

Choi, W., Hong, J.W.-K., 2021. Performance evaluation of ethereum private and testnet networks using hyperledger caliper. In: 2021 22nd Asia-Pacific Network Operations and Management Symposium. APNOMS, IEEE, pp. 325–329.

Coelho, J., Valente, M.T., 2017. Why modern open source projects fail. In: Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering. Association for Computing Machinery, New York, NY, USA, pp. 186–196.

Constantino, K., Souza, M., Zhou, S., Figueiredo, E., Kästner, C., 2023. Perceptions of open-source software developers on collaborations: An interview and survey study. J. Softw.: Evol. Process. 35 (5), e2393.

Curto-Millet, D., Jiménez, A.C., 2023. The sustainability of open source commons. Eur. J. Inf. Syst. 32 (5), 763–781.

Dijkers, J., Sincic, R., Wasankhasit, N., Jansen, S., 2018. Exploring the effect of software ecosystem health on the financial performance of the open source companies. In: Proceedings of the 1st International Workshop on Software Health. SoHeal '18, Association for Computing Machinery, New York, NY, USA, pp. 48–55.

Fan, Y., Xiao, T., Hata, H., Treude, C., Matsumoto, K., 2024a. "My GitHub sponsors profile is live!" investigating the impact of Twitter/X mentions on GitHub sponsors. In: Proceedings of the IEEE/ACM 46th International Conference on Software Engineering. ICSE '24, Association for Computing Machinery, New York, NY, USA.

Fan, Y., Xiao, T., Hata, H., Treude, C., Matsumoto, K., 2024b. "My GitHub sponsors profile is live!" investigating the impact of Twitter/X mentions on GitHub sponsors. In: Proceedings of the 46th IEEE/ACM International Conference on Software Engineering, ICSE 2024, Lisbon, Portugal, April 14–20, 2024. ACM, pp. 191:1–191:12.

Foucault, M., Palyart, M., Blanc, X., Murphy, G.C., Falleri, J.-R., 2015. Impact of developer turnover on quality in open-source software. In: Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering. ACM, Bergamo Italy, pp. 829–841.

Gamalielsson, J., Lundell, B., 2012. Long-term sustainability of open source software communities beyond a fork: A case study of LibreOffice. In: Hammouda, I., Lundell, B., Mikkonen, T., Scacchi, W. (Eds.), Open Source Systems: Long-Term Sustainability. In: IFIP Advances in Information and Communication Technology, Springer, Berlin, Heidelberg, pp. 29–47.

Ge, H., Zhao, L., Yue, D., Xie, X., Xie, L., Gorbachev, S., Korovin, I., Ge, Y., 2024. A game theory based optimal allocation strategy for defense resources of smart grid under cyber-attack. Inf. Sci. 652, 119759.

Gervais, A., Karame, G.O., Wüst, K., Glykantzis, V., Ritzdorf, H., Capkun, S., 2016. On the security and performance of proof of work blockchains. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. CCS '16, Association for Computing Machinery, New York, NY, USA, pp. 3–16.

Hann, I.-H., Roberts, J.A., Slaughter, S.A., 2013. All are not equal: An examination of the economic returns to different forms of participation in open source software communities. Inf. Syst. Res. 24 (3), 520–538.

Ho, S.Y., Richardson, A., 2013. Trust and distrust in open source software development. J. Comput. Inf. Syst. 54 (1), 84–93.

Homscheid, D., Schaarschmidt, M., 2016. Between organization and community: investigating turnover intention factors of firm-sponsored open source software developers. In: Proceedings of the 8th ACM Conference on Web Science, WebSci 2016, Hannover, Germany, May 22–25, 2016. ACM, pp. 336–337.

Howison, J., Herbsleb, J.D., 2013. Incentives and integration in scientific software production. In: Proceedings of the 2013 Conference on Computer Supported Cooperative Work. CSCW '13, Association for Computing Machinery, New York, NY, USA, pp. 459–470.

Hu, K., Zhu, J., Ding, Y., Bai, X., Huang, J., 2020. Smart contract engineering. Electronics 9 (12), 2042.

Iaffaldano, G., Steinmacher, I., Calefato, F., Gerosa, M., Lanubile, F., 2019a. Why do developers take breaks from contributing to OSS projects? A preliminary analysis. In: Proceedings of the 2nd International Workshop on Software Health. Montreal, Quebec, Canada, pp. 9–16.

Iaffaldano, G., Steinmacher, I., Calefato, F., Gerosa, M., Lanubile, F., 2019b. Why do developers take breaks from contributing to OSS projects? A preliminary analysis. In: Proceedings of the 2nd International Workshop on Software Health. SoHeal '19, IEEE Press, pp. 9–16.

Jahn, L., Engelbutzeder, P., Randall, D., Bollmann, Y., Ntouros, V., Michel, L.K., Wulf, V., 2024. In between users and developers: Serendipitous connections and intermediaries in volunteer-driven open-source software development. In: Proceedings of the CHI Conference on Human Factors in Computing Systems. CHI '24, Association for Computing Machinery, New York, NY, USA.

Jamieson, J., Yamashita, N., Foong, E., 2024. Predicting open source contributor turnover from value-related discussions: An analysis of GitHub issues. In: Proceedings of the IEEE/ACM 46th International Conference on Software Engineering. ICSE '24, Association for Computing Machinery, New York, NY, USA.

Joo, C., Kang, H., Lee, H., 2012. Anatomy of open source software projects: Evolving dynamics of innovation landscape in open source software ecology. In: The 5th International Conference on Communications, Computers and Applications, MIC-CCA2012. pp. 96–100.

Kirchenbauer, J., Geiping, J., Wen, Y., Katz, J., Miers, I., Goldstein, T., 2023. A watermark for large language models. In: International Conference on Machine Learning. PMLR, pp. 17061–17084.

Lamprecht, A.-L., Garcia, L., Kuzak, M., Martinez, C., Arcila, R., Martin Del Pico, E., Dominguez Del Angel, V., Van De Sandt, S., Ison, J., Martinez, P.A., et al., 2020. Towards FAIR principles for research software. Data Sci. 3 (1), 37–59.

Lazear, E.P., Rosen, S., 1981. Rank-Order Tournaments as Optimum Labor Contracts. J. Political Econ. 89 (5), 841–864.

Leblang, D., Smith, M.D., Wesselbaum, D., 2022. The effect of trust on economic performance and financial access. Econom. Lett. 220, 110884.

Li, X., 2021. An anti-tampering model of sensitive data in link network based on blockchain technology. Web Intell. 19 (3), 227–237.

Li, Z., Wang, C., Wang, S., Gao, C., 2023. Protecting intellectual property of large language model-based code generation APIs via watermarks. In: Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security. CCS '23, Association for Computing Machinery, New York, NY, USA, pp. 2336–2350.

Li, Z., Yu, Y., Wang, T., Yin, G., Li, S., Wang, H., 2022. Are you still working on this? An empirical study on pull request abandonment. IEEE Trans. Softw. Eng. 48 (6), 2173–2188.

Liao, Z., Song, S., Zhu, H., Luo, X., He, Z., Jiang, R., Chen, T., Chen, J., Zhang, T., Zhang, X., 2023. Large-scale empirical study of inline assembly on 7.6 million ethereum smart contracts. IEEE Trans. Softw. Eng. 49 (2), 777–801.

Liu, L., Tang, C., Zhang, L., Liao, S., 2024. A generic approach for network defense strategies generation based on evolutionary game theory. Inf. Sci. 677, 120875.

Luu, L., Chu, D.-H., Olickel, H., Saxena, P., Hobor, A., 2016. Making smart contracts smarter. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. pp. 254–269.

Miller, M., Doerr, G., Cox, I., 2004. Applying informed coding and embedding to design a robust high-capacity watermark. IEEE Trans. Image Process. 13 (6), 792–807.

Miller, C., Widder, D.G., Kästner, C., Vasilescu, B., 2019. Why do people give up FLOSSing? A study of contributor disengagement in open source. In: Open Source Systems - 15th IFIP WG 2.13 International Conference, OSS 2019, Montreal, QC, Canada, May 26–27, 2019, Proceedings. In: IFIP Advances in Information and Communication Technology, vol. 556, Springer, pp. 116–129.

Moe, N.B., Šmite, D., 2008. Understanding a lack of trust in global software teams: a multiple-case study. Softw. Process.: Improv. Pr. 13 (3), 217–231.

Monrat, A.A., Schelén, O., Andersson, K., 2019. A survey of blockchain from the perspectives of applications, challenges, and opportunities. IEEE Access 7, 117134–117151.

Nakasai, K., Hata, H., Onoue, S., Matsumoto, K., 2017. Analysis of donations in the eclipse project. In: 2017 8th International Workshop on Empirical Software Engineering in Practice. IWESEP, pp. 18–22.

Nguyen, H.-L., Ignat, C.-L., Perrin, O., 2018. Trusternity: Auditing transparent log server with blockchain. In: Companion Proceedings of the the Web Conference 2018. WWW '18, International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, pp. 79–80.

Oláh, J., Hidayat, Y.A., Popp, J., Lakner, Z., Kovács, S., 2021. The effect of integrative trust and innovation on financial performance in a disruptive era. Econ. Sociol. 14 (2), 111–136.

Overney, C., 2020. Hanging by the thread: an empirical study of donations in open source. In: Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Companion Proceedings. ICSE '20, Association for Computing Machinery, New York, NY, USA, pp. 131–133.

Overney, C., Meinicke, J., Kästner, C., Vasilescu, B., 2020. How to not get rich: an empirical study of donations in open source. In: Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering. ICSE '20, Association for Computing Machinery, New York, NY, USA, pp. 1209–1221.

Pacheco, M., Oliva, G.A., Rajbahadur, G.K., Hassan, A.E., 2023. Is my transaction done yet? An empirical study of transaction processing times in the ethereum blockchain platform. ACM Trans. Softw. Eng. Methodol. 32 (3), 59:1–59:46.

Panigrahi, R., Kuanar, S.K., Kumar, L., 2022. Cross-project software refactoring prediction using optimized deep learning neural network with the aid of attribute selection. Int. J. Open Source Softw. Process. 13 (1), 1–31.

Patrickson, B., 2021. What do blockchain technologies imply for digital creative industries? Creativity Innov. Manag. 30 (3), 585–595.

Pi, S.-M., Liao, H.-L., Chen, H.-M., 2012. Factors that affect consumers' trust and continuous adoption of online financial services. Int. J. Bus. Manag. 7 (9), 108.

Qiu, H.S., Li, Y.L., Padala, S., Sarma, A., Vasilescu, B., 2019a. The signals that potential contributors look for when choosing open-source projects. Proc. ACM Hum.- Comput. Interact. 3 (CSCW).

Qiu, H.S., Nolte, A., Brown, A., Serebrenik, A., Vasilescu, B., 2019b. Going farther together: the impact of social capital on sustained participation in open source. In: Proceedings of the 41st International Conference on Software Engineering. ICSE '19, IEEE Press, pp. 688–699.

Ren, Y., Huang, D., Wang, W., Yu, X., 2023. BSMD:A blockchain-based secure storage mechanism for big spatio-temporal data. Future Gener. Comput. Syst. 138, 328–338.

Robinson, W.N., Vlas, R.E., 2015. Requirements evolution and project success: An analysis of SourceForge projects. In: 21st Americas Conference on Information Systems, AMCIS 2015, Puerto Rico, August 13–15, 2015. Association for Information Systems.

Samuel, B.M., Bala, H., Daniel, S.L., Ramesh, V., 2022. Deconstructing the nature of collaboration in organizations open source software development: The impact of developer and task characteristics. IEEE Trans. Softw. Eng. 48 (10), 3969–3987.

Schmid, S., Shestakov, D., 2024. Invited paper: Blockchain governance and liquid democracy - quantifying decentralization in gitcoin and internet computer. In: Chatzigiannakis, I., Gramoli, V. (Eds.), Proceedings of the 2024 Workshop on Advanced Tools, Programming Languages, and PLatforms for Implementing and Evaluating Algorithms for Distributed Systems, ApPLIED 2024, Nantes, France, 17 June 2024. ACM, pp. 1–7.

Shimada, N., Xiao, T., Hata, H., Treude, C., Matsumoto, K., 2022. GitHub sponsors: exploring a new way to contribute to open source. In: Proceedings of the 44th International Conference on Software Engineering. ICSE '22, Association for Computing Machinery, New York, NY, USA, pp. 1058–1069.

Silva, J.D.O., Wiese, I.S., German, D.M., Steinmacher, I.F., Gerosa, M.A., 2017. How long and how much: What to expect from summer of code participants? In: 2017 IEEE International Conference on Software Maintenance and Evolution. ICSME, pp. 69–79.

Sun, Z., Du, X., Song, F., Li, L., 2023. CodeMark: Imperceptible watermarking for code datasets against neural code completion models. In: Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering. In: ESEC/FSE 2023, Association for Computing Machinery, New York, NY, USA, pp. 1561–1572.

Tamburri, D.A., Blincoe, K., Palomba, F., Kazman, R., 2020. "The canary in the coal mine..." A cautionary tale from the decline of SourceForge. Softw.: Pr. Exp. 50 (10), 1930–1951.

Tan, X., Zhou, M., Zhang, L., 2023. Understanding mentors' engagement in OSS communities via google summer of code. IEEE Trans. Softw. Eng. 49 (5), 3106–3130.

Tyler, K., Stanley, E., 2007. The role of trust in financial services business relationships. J. Serv. Mark. 21 (5), 334–344.

Valiev, M., Vasilescu, B., Herbsleb, J., 2018a. Ecosystem-level determinants of sustained activity in open-source projects: a case study of the PyPI ecosystem. In: Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. In: ESEC/FSE 2018, Association for Computing Machinery, New York, NY, USA, pp. 644–655.

Valiev, M., Vasilescu, B., Herbsleb, J., 2018b. Ecosystem-level determinants of sustained activity in open-source projects: A case study of the PyPI ecosystem. In: Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. New York, NY, USA, pp. 644–655.

Voong, C., Saebi, S., 2023. If you find yourself in lava, don't panic. Go with the flow! flowing between InnerSource and open source development. In: 1st IEEE/ACM International Workshop on InnerSource Software Development, InnerSoft@ICSE 2023, Melbourne, Australia, May 20, 2023. IEEE, p. 1.

Wang, L., Fu, F., Chen, X., 2024. Mathematics of multi-agent learning systems at the interface of game theory and artificial intelligence. Sci. China Inf. Sci. 67 (6).

Wang, Y., Wang, L., Hu, H., Jiang, J., Kuang, H., Tao, X., 2022. The influence of sponsorship on open-source software developers' activities on GitHub. In: Leong, H.V., Sarvestani, S.S., Teranishi, Y., Cuzzocrea, A., Kashiwazaki, H., Towey, D., Yang, J., Shahriar, H. (Eds.), 46th IEEE Annual Computers, Software, and Applications Conferenc, COMPSAC 2022, Los Alamitos, CA, USA, June 27–July 1, 2022. IEEE, pp. 924–933.

Wermke, D., Wöhler, N., Klemmer, J.H., Fourné, M., Acar, Y., Fahl, S., 2022. Committed to trust: A qualitative study on security & trust in open source software projects. In: 43rd IEEE Symposium on Security and Privacy, SP 2022, San Francisco, CA, USA, May 22–26, 2022. IEEE, pp. 1880–1896.

Xu, H., Wan, J., 2008. Innovation in open source software with knowledge: Three challenges for open source competence centres. In: 2008 4th International Conference on Wireless Communications, Networking and Mobile Computing. pp. 1–4.

Yin, L., Chakraborti, M., Yan, Y., Schweik, C., Frey, S., Filkov, V., 2022. Open source software sustainability: Combining institutional analysis and socio-technical networks. Proc. the ACM Human- Comput. Interact. 6 (CSCW2), 1–23.

Yin, L., Chen, Z., Xuan, Q., Filkov, V., 2021. Sustainability forecasting for apache incubator projects. In: Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. In: ESEC/FSE 2021, Association for Computing Machinery, New York, NY, USA, pp. 1056–1067.

Yu, Y., Benlian, A., Hess, T., 2012. An empirical study of volunteer members' perceived turnover in open source software projects. In: 2012 45th Hawaii International Conference on System Sciences. IEEE, pp. 3396–3405.

Zahedi, M., Babar, M.A., Cooper, B., 2018. An empirical investigation of transferring research to software technology innovation: a case of data-driven national security software. In: Oivo, M., Fernández, D.M., Mockus, A. (Eds.), Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM 2018, Oulu, Finland, October 11–12, 2018. ACM, pp. 10:1–10:10.

Zanetti, M.S., 2012. The co-evolution of socio-technical structures in sustainable software development: Lessons from the open source software communities. In: 2012 34th International Conference on Software Engineering. ICSE, pp. 1587–1590.

Zantalis, F., Koulouras, G.E., Karabetsos, S., 2024. Blockchain technology: A framework for endless applications. IEEE Consum. Electron. Mag. 13 (2), 61–71.

Zhang, L., Choffnes, D., Levin, D., Dumitraş, T., Mislove, A., Schulman, A., Wilson, C., 2014. Analysis of SSL certificate reissues and revocations in the wake of heartbleed. In: Proceedings of the 2014 Conference on Internet Measurement Conference. pp. 489–502.

Zhang, Y., Qin, M., Stol, K.-J., Zhou, M., Liu, H., 2024. How are paid and volunteer open source developers different? A study of the rust project. In: Proceedings of the IEEE/ACM 46th International Conference on Software Engineering. ICSE '24, Association for Computing Machinery, New York, NY, USA.

Zhang, X., Wang, T., Yu, Y., Zeng, Q., Li, Z., Wang, H., 2022. Who, what, why and how? Towards the monetary incentive in crowd collaboration:A case study of github's sponsor mechanism. In: Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems. CHI '22, Association for Computing Machinery, New York, NY, USA, pp. 1–18.

Zhang, Z., Yang, Y., He, H., Chen, J., 2023. An empirical study on GitHub sponsor mechanism. Int. J. Softw. Eng. Knowl. Eng. 33 (9), 1439–1465.

Zheng, Z., Xie, S., Dai, H.-N., Chen, W., Chen, X., Weng, J., Imran, M., 2020. An overview on smart contracts: Challenges, advances and platforms. Future Gener. Comput. Syst. 105, 475–491.

Zhou, L., Tang, C., Bao, Z., Liu, Y., Yu, X., 2024. A reputation-based blockchain scheme for sustained carbon emission reduction. Sci. China Inf. Sci. 67 (5).

Zhou, J., Wang, S., Bezemer, C.-P., Zou, Y., Hassan, A.E., 2021. Studying the association between bountysource bounties and the issue-addressing likelihood of GitHub issue reports. IEEE Trans. Softw. Eng. 47 (12), 2919–2933.

Zhou, J., Wang, S., Kamei, Y., Hassan, A.E., Ubayashi, N., 2021a. Studying donations and their expenses in open source projects: a case study of GitHub projects collecting donations through open collectives. Empir. Softw. Eng. 27 (1), 24.

Zhou, J., Wang, S., Kamei, Y., Hassan, A.E., Ubayashi, N., 2022. Studying donations and their expenses in open source projects: a case study of GitHub projects collecting donations through open collectives. Empir. Softw. Eng. 27 (1), 24.

Zhou, J., Wang, S., Zhang, H., Chen, T.P., Hassan, A.E., 2021b. Studying backers and hunters in bounty issue addressing process of open source projects. Empir. Softw. Eng. 26 (4), 81.

Zou, W., Lo, D., Kochhar, P.S., Le, X.-B.D., Xia, X., Feng, Y., Chen, Z., Xu, B., 2021. Smart contract development: Challenges and opportunities. IEEE Trans. Softw. Eng. 47 (10), 2084–2106.