A Multidimensional Contract Design for Smart Contract-as-a-Service

Jinghan Sun, Hou-Wan Long, Hong Kang, Zhixuan Fang, *Member, IEEE*, Abdulmotaleb El Saddik, *Fellow, IEEE*, and Wei Cai, *Senior Member, IEEE*

***Abstract*—Empowered by blockchain technology, smart contracts have attracted considerable interest from Web3 users due to their distinct advantages. Nevertheless, it is challenging to address problems caused by the dramatic expansion of the Web3 ecosystem. This article introduces the smart contract-as-a-service (SCaaS) paradigm to mitigate smart contracts' redundant deployment via their composability and reusability. Moreover, we design trust and incentive schemes to ensure project security and developer engagement in SCaaS. Specifically, we first introduce a reputation filter by leveraging the authentic on-chain data, aiming to eliminate high-risk contracts. We then design a contract-based incentive mechanism to help the foundation attract heterogeneous developers with multidimensional private information, and maximize the foundation's utility by inducing developers to undertake projects of differing complexities based on their ability. We further differentiate between veteran and newcome developers and examine their influences on foundational strategies. Finally, extensive experimental results demonstrate that our proposed contracts can efficiently remove high-risk smart contracts, maximize the foundation's utility, and ensure that developers select contracts honestly and participate in the SCaaS ecosystem actively.**

***Index Terms*—Incentive mechanism (IM), multidimensional contract, reputation evaluation, smart contract-as-a-service (SCaaS).**

I. Introduction

BLOCKCHAIN technology endows smart contracts with decentralization and immutability, thereby enabling smart contracts attracting a large number of users and developers to participate in the Web3 ecosystem currently [1], [2], [3]. However, the sharp expansion of the ecosystem has caused various problems that hinder Web3 development. First, the rapid surge of on-chain smart contracts' deployment reduces blockchain's decentralization. Using Ethereum (ETH) as an example, the number of on-chain smart contracts has escalated 11 times in the past year[1], thus leading to the size of Ethereum's full nodes expanding from 2.0 terabytes in January 2019 to 16.9 terabytes in January 2024[2]. This excessive data storage imposes a higher threshold for new nodes' participation in the blockchain, consequently diminishing the decentralization of the blockchain [4]. Second, users' increasingly complex demands and higher quality expectations for decentralized applications (Dapps) lead to their elevated development costs. Users now require that smart contracts not only be more secure but also perform more efficiently, minimizing vulnerabilities and reducing transaction costs [5]. Additionally, they expect Dapps to seamlessly integrate with existing Web2 applications, allowing for a smoother and more interconnected user experience [6]. Furthermore, the fierce competition in the Web3 market significantly augments user acquisition costs for Dapps to vie for user engagement and retention [7]. At the same time, the security risks associated with Dapps intensify significantly. From January 2023 to January 2024, the number of unique active wallets (UAE)[3] on the Ethereum Virtual Machine (EVM) surged by more than 5.7 times[4], accompanied by the number rise in malicious wallets. Additionally, the escalation in both the quantity and complexity of Dapps presents challenges in scrutinizing and managing smart contracts, thereby jeopardizing the financial security of Web3 users due to the vulnerabilities inherent in these contracts [8].

To tackle these challenges, it is necessary to encourage developers to reuse the existing smart contracts with target

Received 5 March 2024; revised 25 October 2024; accepted 4 November 2024. This work was supported in part by Guangdong Basic and Applied Basic Research Foundation under Grant 2024A1515012323; in part by the Open Topics of Key Laboratory of Blockchain Technology and Data Security, The Ministry of Industry and Information Technology of the People's Republic of China; and in part by the CUHK(SZ)-White Matrix Joint Metaverse Laboratory. *(Corresponding author: Wei Cai.)*

Jinghan Sun is with The Chinese University of Hong Kong, Shenzhen 518172, China, and also with Mohamed Bin Zayed University of Artificial Intelligence (MBZUAI), Masdar City, UAE (e-mail: Jinghansun@link.cuhk.edu.cn).

Hou-Wan Long is with The Chinese University of Hong Kong, Hong Kong, China (e-mail: houwanlong@link.cuhk.edu.hk).

Hong Kang and Wei Cai are with The Chinese University of Hong Kong, Shenzhen 518172, China (e-mail: hongkang1@link.cuhk.edu.cn; caiwei@cuhk.edu.cn).

Zhixuan Fang is with Tsinghua University, Beijing 100084, China, and also with Shanghai Qi Zhi Institute, Shanghai 200232, China (e-mail: zfang@mail.tsinghua.edu.cn).

Abdulmotaleb El Saddik is with Mohamed Bin Zayed University of Artificial Intelligence (MBZUAI), Masdar City, UAE and also with the University of Ottawa, Ottawa, ON K1N 6N5, Canada (e-mail: elsaddik@uottawa.ca).

Digital Object Identifier 10.1109/TCSS.2024.3514924

[1]https://dune.com/sharptraderx/scdot

[2]https://etherscan.io/chartsync/chainarchive

[3]A Unique Active Wallet refers to singular crypto wallet address that interacts with Dapp's smart contracts.

[4]https://dune.com/queries/2759476/4591722

functionalities during Dapps development. This approach can mitigate the storage burden by avoiding on-chain smart contracts' duplication, save marginal costs in both development and user acquisition, and bolster project security by reusing mature and trustworthy smart contract modules. However, the situation of reusing smart contracts is not that optimistic. Over 60% of smart contracts have never been executed, and 90% of smart contracts have been executed less than ten times [9]. This issue is partly due to the potential transfer of security vulnerabilities from the existing projects to new projects [10], and the inadequacy of effective incentive mechanisms. Existing public chains and platforms mainly encourage developers to reuse smart contracts through two indirect incentives: simplifying the reuse process and improving developers' abilities. Yet, no public chains or platforms offer direct incentives for smart contracts' reuse, such as distributing bonuses, highlighting a gap in encouraging smart contract reuse effectively.

On the other hand, as a far-reaching official organization that encourages developers' engagement through direct incentive[5], the blockchain foundation has no encouragement for reusing smart contracts during this process. However, based on the advantages of reusing smart contracts, foundations have sufficient motivation to promote this practice. Additionally, the project proposals are screened by foundation staff and the selection criteria are ambiguous, which is overly centralized and introduces potential bias and opaque competition. This approach may deter developer involvement in ecosystem development [11], [12]. Moreover, it is challenging for the foundation to design the incentive mechanism. Since the escalating complexity of Dapps leads to increased development costs, it is crucial to offer rewards that match the complexity of projects to encourage developers to tackle sophisticated tasks that meet market demands. However, developers with lower ability are not capable of handling higher complexity in smart contracts [13]. Concurrently, developers' private information, i.e., their abilities and the reputation of their projects, remains undisclosed. Developers may tend to hide their information to gain higher profits, making it challenging to induce them to disclose their private information during the incentive process.

To help the foundation overcome the aforementioned shortages, this article introduces a three-stage system, as depicted in Fig. 1. We first propose the Smart contract-as-a-service (SCaaS) paradigm to mitigate the smart contract's redundant deployment. Within this service-oriented computing model, developers can use smart contracts as computing components to achieve smart contracts' reusability and composability, streamlining Dapp development by assembling existing contracts such as "LEGO bricks". The SCaaS ecosystem comprises two key roles: 1) a public blockchain foundation that puts forward project requirements, and 2) a group of developers willing to develop these projects. Specifically, the public blockchain foundation declares its desired project types, and developers submit proposals about what existing smart contracts they plan to reuse by SCaaS, adhering to the foundation's demands. These proposals will be screened in the next stage to decide developers that can engage in the incentive stage.

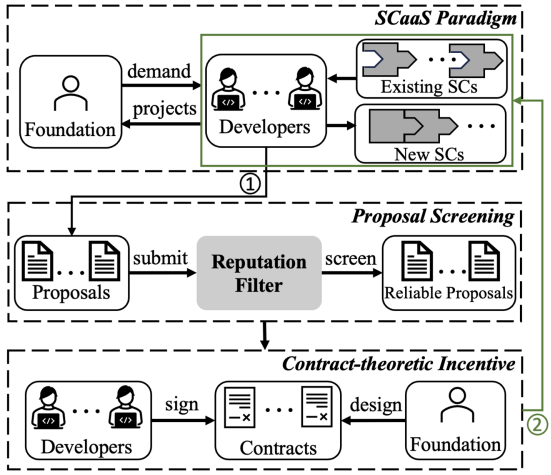


Fig. 1. Framework of the system.

In the next stage, we introduce a "reputation filter" to replace the centralized proposal screening process run by the foundation staff. Inspired by the research [14], [15], which indicate the significant impact of project reputation on user behavior, the reputation filter takes the reputation of reused smart contracts as a key screening criterion to deter malicious developers from reusing unsafe smart contracts. By leveraging the traceability of on-chain transactions, the reputation ratings are evaluated by using transaction volume as the calculation metric. Contracts with low reputations are considered "high-risk" or "malicious" and are excluded, while those passing the filter are considered trustworthy, eligible to participate in the incentive stage, and entitled to appropriate rewards.

In the final stage, to attract developers using SCaaS to deploy smart contracts according to their proposal, we help the foundation design a contract-theoretic incentive mechanism. Contract theory is particularly relevant in a monopolized market setting, where there is an employer (i.e., the foundation) and a group of employees (i.e., developers). The foundation, unaware of developers' private information, can overcome this information asymmetry by presenting a list of contracts tailored for heterogeneous developers, which is a core feature of contract theory [16], [17]. While the foundation does not know developers' private information, he can overcome the information asymmetry by providing a contracts list for heterogeneous developers based on contract theory. These contracts offer projects of varying complexity tailored to developers' different levels of abilities and ensure that developers honestly select the contract specifically designed for them based on their private information. This approach aims to provide feasible payoffs for developers while maximizing the foundation's profit.

[5]Using the BNB Chain Builder Grants and the ETH Ecosystem Support Project as illustrations, these initiatives define their focus areas and encourage developers to submit proposals that align with them. After evaluating these proposals, the organizations reward winners with native tokens, stablecoins, or fiat currency, supporting further development.

Specifically, we aim to achieve the following targets at the incentive stage: 1) Identification of the conditions that motivate developers to participate in projects. Most existing literature assumes developers' eagerness to engage, but this idealistic assumption overlooks developer involvement's cost. It's necessary to design the incentive mechanism for a more realistic scenario. 2) Incentive mechanism design for multidimensional private information. While most studies consider only one-dimensional private information, multiple factors influence the benefits for participants. Both developers' ability and projects' reputation impact the foundation's revenue, thereby requiring simultaneous consideration of these factors in the incentive stage. 3) Analysis of the effects of developer population variability on foundation strategy. The literature usually assumes a fixed amount of developers during decision-making. In this article, by introducing the "reputation filter", the reputation threshold alters the number of developers eligible for the incentive stage, thus complicating the identification of the foundation's optimal strategy. 4) Evaluation of newcome and veteran developers' influence on contract formulation. Prior collaboration experience among developers and the foundation influences the foundation's understanding of developers' private information. It's necessary to analyze how this varying private information understanding influences the foundation's contract design, as well as its strategies employed to retain veteran developers while attracting new ones.

Our major contributions are summarized as follows:

1) *Interaction Characterization with Multidimensional Information in SCaaS Market.* We build the model that captures the foundation and developers' interactions based on realistic scenarios involving information asymmetry within the SCaaS market. To the best of our knowledge, we are the first to propose the SCaaS paradigm and conduct the related economic analysis.
2) *Reputation Filter Design Based on the On-chain Data.* We propose a reputation filter to ensure developers reuse trustworthy smart contracts. Unlike most existing work, this approach leverages authentic on-chain data as a criterion, thus ensuring the reused smart contract's reliability and escalating the cost of malicious behavior.
3) *Incentive Mechanism Design Under Multidimensional Private Information.* We design incentive contracts for developers based on multidimensional private information by utilizing contract theory. We analyze the optimal contract design strategy and investigate the impact of newcome and veteran developers on the optimal contracts.
4) *Performance Evaluation of the Incentive Contract.* We corroborate the effectiveness of our proposed incentive mechanism through extensive simulations. Experimental results show that the foundation should set an optimal reputation threshold to both ensure project reliability and foster wider developer engagement. Our optimal contracts stimulate developers with higher abilities to engage in the SCaaS ecosystem, promoting honest contract selection and the undertaking of more complex projects.

II. Related Work

A. As-a-Service Paradigm

The "as-a-Service" paradigm, a cornerstone of cloud computing, can be defined as a pay-as-you-go model that offers users web-based services utilizing cloud computing resources [18], [19], [20], [21]. This model encompasses widely recognized services such as Software-as-a-Service (SaaS) [18], Platform-as-a-Service (PaaS) [19] and, Infrastructure-as-a-Service (IaaS) [20], [21]. Beyond these traditional cloud computing services, Blockchain-as-a-Service (BaaS) merges the cloud computing service and blockchain technology, aiming to facilitate the construction of new blockchains [22]. However, BaaS targets building new blockchains, rather than providing smart contract services. Expanding on this model, several new frameworks have emerged to integrate blockchain with other critical systems. Notable examples include the blockchain-enhanced Sensor-as-a-Service (SEaaS) [23], which enhances IoT systems security, the Integrated Fuzzy Decision Tree based Blockchain Federated Safety-as-a-Service for IIoT (IFDT-BCF-SAS-IIOT) [24], which focuses on enhancing both IoT environment security and decision accuracy, the Trusted Computing-as-a-Service (TCaaS) [25], which strengthens data confidentiality. In parallel, smart contracts—automated codes stored on a blockchain—have gained considerable attention due to their immutable nature, providing decentralized services without requiring a trusted external authority. The literature has considered its benefits and use of smart contracts to provide services in different areas, such as blockchain-driven Metaverse (e.g., Duan et al. [26]), federated learning (e.g., Fan et al. [27]), and cloud gaming (e.g., Fan et al. [28]), etc. To the best of our knowledge, our article is the first to propose the novel SCaaS paradigm that utilizes smart contracts as a computing component to provide service based on reusing smart contracts.

B. Incentive Scheme in Smart Contracts Reuse

Current practices and literature support developers in reusing smart contracts through two approaches. On the one hand, they focus on reducing the barriers to smart contract reuse. Initiatives such as OpenZeppelin[6] and Gnosis V2[7] provide open-source smart contract repositories on GitHub, offering developers standardized, audited, and reusable smart contracts. Part et al. [29] and Khan et al. [30] proposed a mechanism to standardize the management and sharing of smart contracts, enhancing their reusability. Moreover, development tools such as Truffle[8] convert the complex raw requests made to the ETH Network into contract abstractions, streamlining the process of smart contract reuse. Shen et al. [31] proposed a visualization approach to generate a visual solidity code to make it easier to reuse smart contracts. Hsian et al. [32] introduced a graphical editor designed to standardize protocols, reducing the complexity of migrating contracts across different platforms and blockchains.

[6]https://github.com/OpenZeppelin/openzeppelin-contracts
[7]https://github.com/gnosis/gp-v2-contracts
[8]https://trufflesuite.com/docs/truffle/how-to/contracts

TABLE I
MOSTLY USED SMART CONTRACTS BASED ON EVM BLOCKCHAIN

Rank	Contract Address	Project Name	Transactions Amount
1	0x00000000006c3852cbef3e08e8df289169ede581	OpenSea	12629208
2	0x68b3465833fb72a70ecdf485e0e4c7bd8665fc45	UniSwap V3	10852857
3	0xef1c6e67703c7bd7107eed8303fbe6ec2554bf6b	Uniswap Old Version	10493008
4	0x7a250d5630b4cf539739df2c5dacb4c659f2488d	Uniswap Deployer	10024286
...	...	...	...
999	0xd54f502e184b6b739d7d27a6410a67dc462d69c8	dYdX: Gps Statement Verifier Governor	20738
1000	0x4dd942baa75810a3c1e876e79d5cd35e09c97a76	Dash 2 Trade	20713
...	...	...	...
2965	0xc068efbf9009dcf06a5783f8bb8cac66311dfab5	DrunkenVikingsTribe	6000

Meanwhile, there's an emphasis on enhancing developers' skills in smart contract reuse. Developers are encouraged to participate in discussions and seek advice on smart contract development, including aspects of reusability, on platforms such as Ethereum Stack Exchange[9] and Solidity Forum[10], enabling developers to reuse smart contracts more easily.

C. Reputation Evaluation Scheme

The reputation model, extensively utilized in P2P (peer-to-peer) networks and e-commerce scenarios [33], reflects the perspectives and evaluations between the task publisher and the developers, or between developers themselves, in the crowdsourcing process. Reputation value is primarily gauged through trust rating [33], [34]. In terms of trust rating calculation, the existing literature primarily classifies trust into direct trust and indirect trust [35], [36], [37]. Direct trust originates from personal experience, referring to the trust the task publisher establishes from the target developer based on their interactions and experiences. Indirect trust arises from third-party recommendations, indicating the trust other developers or users form from the target developer. Parhizka et al. [35] and Dai et al. [37] calculated trust ratings by using the binary rating method, where negative ratings signify distrust and positive ones convey trust. Su et al. [38] utilized multiscale rating to calculate diverse degrees of trust. Subsequently, the total trust rating, indicative of the overall reputation, is determined by a sum of both direct and indirect trust, considering their respective weight factors.

III. REPUTATION FILTER

A. Aim of the Filter

To bolster the security of smart contracts within the SCaaS paradigm, we first verify the reliability of the smart contracts intended for reuse. A reputation filter is thus established before the incentive phase, aiming at eliminating risky smart contracts. This process involves assessing the smart contract's risk by calculating the reputation value of the existing smart contracts it reuses. Nonetheless, conventional methods for calculating reputation prove inadequate for Dapps. In the SCaaS scenario, direct trust represents the foundation's trust rating against existing smart contracts that developers intend to reuse, and this is also how public blockchain selects project proposals in reality. Yet, this approach is highly centralized, granting the foundation excessive control over proposal selection. As a result, we do not consider direct trust in our system. Conversely, indirect trust represents Web3 users' trust ratings placed in the reused smart contracts of the new projects. However, this method is more vulnerable to Sybil attacks in decentralized networks, where malicious accounts can more effortlessly endorse risky smart contracts, leading to manipulated outcomes. Hence, we introduce the following approach for calculating indirect trust.

B. Reputation Filter Model

Compared with centralized networks, decentralized networks have a unique characteristic: every on-chain transaction requires a gas fee. Consequently, if the volume of on-chain transactions for smart contracts serves as a criterion for calculating indirect trust, then engaging in malicious behavior would require incurring gas fees to manipulate the indirect trust ratings. Considering that the ETH foundation builder grants cap at 30 000 USDs, with the evaluation that the average gas fee per transaction on the ETH last year was around 5 USDs[11], we include all smart contracts exceeding 6000 transactions in our reputation repository, shown in Table I, and deem them relatively trustworthy. In this framework, for a malicious account to get a risky smart contract listed in the repository, it would need to spend a minimum of 30 000 USDs on gas fees, thereby making the cost of malicious activities surpass the maximum reward available for development participation. In this table, we observe that OpenSea ranks as the most utilized in the table[12], with DrunkenVikingTribe being the least. Therefore, the reputation value of new smart contracts generated through the SCaaS paradigm can be defined as

$$\tau_{(i,n)} = \frac{\sum_{l=1}^{L} \tau_l}{L}, L \neq 0 \quad (1)$$

where $\tau_{(i,n)}$ denotes the reputation value of developer n of type-i, as outlined in Section IV-B, and τ_l signifies the reputation value of the existing smart contract proposed for reuse

[9]https://ethereum.stackexchange.com/
[10]https://forum.soliditylang.org/
[11]https://dune.com/lucadavid049/gas-station-20
[12]Actually, USDT records the highest volume of on-chain transactions, but as it is a stablecoin not satisfying the specific demands of project development, we have opted to exclude it.

by the developer, with L representing the total amount of smart contracts being reused ($\tau_{(i,n)}$ equals zero when a developer opts not to reuse any smart contract). To establish the value of τ_l, we assign the transaction count of the highest-ranked smart contract a baseline value of 1. This method facilitates the calculation of the reputation value for each reused smart contract in the library (represented as τ_l) by dividing its transaction volume by that of the benchmark contract. Consequently, we have $\tau_{(i,n)} \in [0, 1]$, and only those with $\tau_{(i,n)} \geq \tau_{\min}$ are permitted through the reputation filter, where $\tau_{\min}$ is the reputation threshold for a submitted smart contract to progress to the subsequent incentive stage.

Unlike most existing work which develops a third-party platform to compute new projects' reputation value or relies on the foundation staff, we leverage the authentic on-chain data as the formulation metrics to accomplish the same objective more efficiently and reliably.

IV. System Model

A. Use Case Example

The complexity of Dapp smart contracts development is intimately related to their functionalities. Taking DeFi (decentralized finance) projects as an example, the market presents a range from Dapps that only facilitate basic swap operations to those offering sophisticated features such as lending and leveraging. We classify the market's trending DeFi projects according to different complexity tiers, each with distinct characteristics.

1) *Low Complexity DeFi*: These are fundamental token swap and exchange platforms, such as Uniswap V1 and MetaMask, dedicated solely to token exchanges.
2) *Medium Complexity DeFi*: Lending and borrowing platforms, such as Aave, require more complex smart contracts to manage collateral and interest rates. Another project named "blend" extends support to both ERC20 token and NFT collaterals, necessitating the development of more complex smart contracts for integration of DeFi and NFT.
3) *High Complexity DeFi*: This tier includes advanced decentralized exchanges such as dydx, which not only facilitate token exchanges and lending but also extend services to leverage and derivatives.

Projects with lower complexity have lower development cost and are more user-friendly, making them suitable for Web3 market beginners, while high complexity projects demand more development expense but cater to the demands of professional users. To meet users' diverse trading demands, foundations should incentivize developers to develop DeFi projects spanning these different complexity levels. Inspired by [39], which shows that developers with greater skills can handle more complex tasks due to their growing experience, and [40], which demonstrates that higher rewards motivate developers to tackle more complex projects, we assume developers with higher ability are capable of handling higher complexity in smart contracts and receive more rewards accordingly. The purpose of this section is to help the blockchain foundation design a series of contracts. Within these contracts, developers, categorized by varying ability levels (denoted as θ) select smart contract development tasks of corresponding complexity (denoted as a). Furthermore, under the premise of ensuring the foundation's revenue, these developers will receive rewards that align with the complexity of their respective projects.

B. Developer's Type

We consider that the developers are within a monopolistic marketplace of the SCaaS rewarding ecosystem, where there is only one public blockchain foundation[13]. We assume that a population of N developers submitting proposals to the reputation filter. Developers are characterized by two-dimensional information: the development ability θ and the reputation value τ of their smart contracts. Within this framework, a developer n ($n \in \{1, N\}$) with $\Gamma_i \triangleq (\theta_i, \tau_{(i,n)})$ is defined as a type-i developer. We assume that all developers fall into to a set $\mathcal{I} = \{1, \ldots, I\}$, consisting of I types. These types are arranged in nondecreasing order of ability θ, i.e., $\theta_1 \leq \theta_2 \leq \cdots \leq \theta_I$. Each type $i \in \mathcal{I}$ comprises N_i developers, and the total number of developers is given by $\sum_{i \in \mathcal{I}} N_i = N$. And $\tau_{(i,n)}$ represents the reputation value of the smart contract associated with developer n, belonging to type-i, and $\tau_{(i,n)}$ is independent of the ability θ. For convenience presentation, we denote $\tau_{(i,n)}$ as the reputation value of a type-i developer.

We posit that during smart contract development, all developers do not change their type. These developers can be further divided into either newcomers or veterans, depending on their prior collaboration experience with the foundation. Veteran developers, having past cooperation with the foundation, are known by the foundation for their private information regarding the developer type. Consequently, the foundation's contract construction for them will be under complete information. In contrast, the foundation lacks prior engagement with new developers, leaving it without knowledge of their private information, and thus involving an incomplete information scenario in contract construction. Nevertheless, the foundation might possess insights into the distribution of developer types based on the market investigation. We propose to design different contracts for veteran and newcome developers, based on symmetric and asymmetric information respectively, to optimize the foundation's profit.

The notations in this article are summarized in Table II.

C. Contract Formulation

Contract theory is a highly utilized theoretical framework investigating how employers design contracts for employees in scenarios with private information asymmetry. In line with this, we introduce a contract-based approach aimed at tackling the specific challenge of designing incentive mechanisms involving the foundation and developers.

[13]According to public blockchains with the largest user and developer base, such as ETH, Polygon, Binance Smart Chain, and Avalanche, each has only one official foundation.

TABLE II
SUMMARY OF NOTATIONS

Symbols	Descriptions
$\tau_{(i,n)}, \tau_l, \tau_{\min}$	The reputation value of developer n's smart contract, the reused smart contract l, and filter's threshold, respectively
L	The number of smart contracts that developer i reuses
θ	The ability of developer
a	The project complexity
n	The individual developer
N	The number of developers submitting proposals
$N^\dagger(\tau_{\min})$	The number of developers passing the filter
N_i	The number of developers that belongs to type-i
Γ	The type of developer
$I, I^\dagger$	The total number of types of developers who submit proposals and pass the reputation filter, respectively
$\mathcal{I}, \mathcal{I}^\dagger$	The set of types of developers who submit proposals and pass the reputation filter, respectively
$\mathcal{L}$	The contract list
Ω	The set of contract items
ω_i	The contract item i
$R(a_i)$	The reward for type-i developer
g_{base}	The basic gas fee of project development
c	The task cost coefficient
α	The satisfaction level of foundation
ψ	The quality coefficient of smart contract
λ_i	The proportion of type-i developers
$U_D^i(\theta_i, \tau_{(i,n)}, \omega_i)$	The utility of type-i developer
$U_F^n(\tau_{\min}, \omega_i), U_F^i(\tau_{\min}, \omega_i)$	The payoff of foundation against developer n and all type-i developers, respectively
$U_F^C(\tau_{\min}, \Omega), U_F^I(\tau_{\min}, \Omega)$	The total payoff of foundation in the scenario involving veteran developers and newcome developers, respectively

A contract list $\mathcal{L} = (\tau_{\min}, \Omega)$ is devised by the foundation to stimulate developers to participate in the SCaaS rewarding system. The contract list describes the relationship between reputation value, smart contracts' complexity, and the corresponding rewards. Within the list, the minimum reputation value $\tau_{\min}$ is a standard requirement for all developer types, acting as a benchmark for the reputation filter. This implies that only developers with a reputation value $\tau_{(i,n)} \geq \tau_{\min}$ fulfill the foundation's criteria and are eligible for rewards, while those with $\tau_{(i,n)} < \tau_{\min}$ are not qualified for signing the contract with the foundation. Besides, the contract list encompasses I contract items $\Omega = \{\omega_i\}_{i \in \mathcal{I}}$, each tailored to a specific developer type. Each contract item $\omega_i \triangleq (a_i, R(a_i))$ defines the relationship between the type-i developer's project complexity and the corresponding reward. Here, a_i represents the required project complexity for each type-i developer in the SCaaS, and $R(a_i)$ indicates the reward, usually the public blockchain's native token, granted to a type-i developer who passes the reputation filter and completes project development. Developers will be out of participation if their payoff, as defined in Section IV-C 1), turns out to be negative.

The contract design is illustrated in Fig. 2. The contract is feasible and optimized if and only if developers choose the contract item which is devised for them deliberately. This means the type-i developer can get the maximized profit only when he chooses the contract item ω_i, completes a smart contract with a project complexity of a_i, and receives a corresponding reward of $R(a_i)$. Specifically, we can ensure the contract's feasibility and optimality by individual rationality (IR) and incentive compatibility (IC) constrain, which we will introduce further in Section V.

1) Payoff Function of Developers: For the type-i developer who signs the contract ω_i, their payoff during the incentive

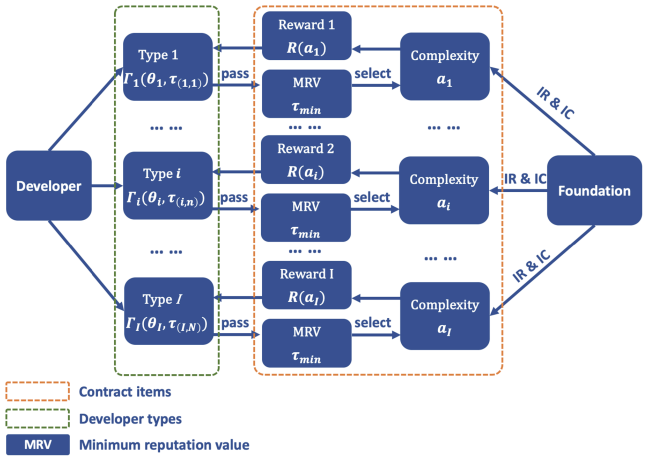


Fig. 2. Contract design.

stage can be defined as the difference between the foundation's reward $R(a_i)$ and the cost incurred in smart contract development. Therefore, type-i developer's payoff can be defined as

$$U_D^i(\theta_i, \tau_{(i,n)}, \omega_i) = \mathbb{1}_{\tau_{(i,n)} \geq \tau_{\min}} R(a_i) - g_{\text{base}} \frac{1}{\theta_i} c a_i \qquad (2)$$

where

$$\mathbb{1}_{\tau_{(i,n)} \geq \tau_{\min}} = \begin{cases} 1, & \text{if} \quad \tau_{(i,n)} \geq \tau_{\min} \\ 0, & \text{Otherwise} \end{cases}$$

Here, c represents the task cost coefficient. Inspired by [33], [41], we assume that each smart contract development incurs a base gas fee, g_{base}, essential for on-chain deployment. And a_i is the project's corresponding complexity, which is proportional to the cost. This cost bears an inverse relationship with the

developer's ability level θ_i, implying that greater developer ability typically results in reduced costs [13]. And $\mathbb{1}_{\tau_{(i,n)}\geq\tau_{\min}}$ represents only developers with $\tau_{(i,n)} \geq \tau_{\min}$ are qualified for the smart contracts development.

2) Payoff Function of Foundation: Under a predetermined minimum reputation value, the foundation's payoff is calculated by subtracting the payment made to developers from the income gained from the project. First, let's consider the foundation's payoff U_F^n obtained from an individual developer n, who belongs to type i and signs the contract ω_i. The payoff function can be defined as follows:

$$U_F^n(\tau_{\min}, \omega_i) = \mathbb{1}_{\tau_{(i,n)}\geq\tau_{\min}}[\alpha \ln(1 + \theta_i a_i \psi + \tau_{\min}) - R(a_i)] \tag{3}$$

where the indicator function $\mathbb{1}_{\tau_{(i,n)}\geq\tau_{\min}}$ signifies that the developer with reputation value $\tau_{(i,n)} < \tau_{\min}$ will be excluded from the incentive stage. Consequently, these developers will neither receive rewards nor contribute to the foundation's profits. $\alpha > 0$ is the foundation's equivalent profit coefficient, reflecting the extent of satisfaction or benefit the foundation gains from a developer's work. Assume that all the developers provide an identical satisfaction coefficient α, with the foundation's profit proportionally increasing as α rises. The smart contract quality coefficient ψ represents the quality of the smart contract. Note that developers with higher ability θ_i can build higher quality projects for the foundation, and projects with higher complexity a_i can attract more users by advanced functionalities, thereby enhancing the foundation's profit. $\tau_{\min}$ is the threshold of the reputation filter that is set by the foundation. Setting a higher $\tau_{\min}$ threshold signals greater trustworthiness to users and can lead to increased market profits for the foundation.

We assume that $\tau_{(i,n)}$ follows a normal distribution, i.e $\tau_{(i,n)} \sim \mathcal{U}(\mu, \sigma^2)$. We use the set $\mathcal{I}^\dagger = \{i | \tau_{(i,n)} \geq \tau_{\min}\}$ to denote the set of developers whose reputation value meets or exceeds $\tau_{\min}$, so the total developer types can be $I^\dagger = |\mathcal{I}^\dagger|$. We then reindex these developer types as $\{i_{\mathcal{I}^\dagger}\}_{i\in\{1,\ldots,I^\dagger\}}$, basing the index solely on the ability θ_i rather than $\tau_{(i,n)}$, since the reputation value does not affect developer's payoff as long as it's no less than $\tau_{\min}$. Given that $\tau_{(i,n)}$ is independent of θ_i, for simplicity, we continue to refer to type-i as type-$i_{\mathcal{I}^\dagger}$. Furthermore, the population of developers eligible for the incentive stage is contingent on the threshold $\tau_{\min}$. Hence, this population can be defined as a function of the minimum reputation value, i.e., $N^\dagger(\tau_{\min})$.

We denote the probability that a developer belongs to type i as λ_i, and $\sum_{i=1}^{I^\dagger} \lambda_i = 1$. Then we can express the foundation's payoff U_F^i against all the type-i developers as

$$\begin{aligned} &U_F^i(\tau_{\min}, \omega_i) \\ &\quad = N^\dagger(\tau_{\min})\lambda_i[\alpha \ln(1 + \theta_i a_i \psi + \tau_{\min}) - R(a_i)]. \end{aligned} \tag{4}$$

Subsequently, we can obtain the foundation's total payoff

$$\begin{aligned} &U_F(\tau_{\min}, \Omega) \\ &\quad = N^\dagger(\tau_{\min}) \sum_{i=1}^{I^\dagger} \lambda_i[\alpha \ln(1 + \theta_i a_i \psi + \tau_{\min}) - R(a_i)]. \end{aligned} \tag{5}$$

In the following, we will categorize developers into two groups: veteran developers and newcome developers, in alignment with realistic scenarios. Subsequently, we will derive optimal strategies tailored to each group's distinct characteristics, utilizing contract theory as the guiding framework.

V. Multidimensional Contract Design

In this section, we analyze the foundation's optimal incentive mechanism for both veteran and newcome developers. For each category, we will first ensure the feasibility of the contract, which means developers can get the maximum payoff when they select the contract deliberately designed for their type. Subsequently, we will make sure the contract is optimal for the foundation which can bring the foundation the highest payoff compared with other feasible contracts.

A. Contract Design for Veteran Developer

The contract design against the veteran developer typically involves scenarios with complete information. Owing to their past interactions or collaborations with the foundation, the foundation is well-informed about each veteran developer's specific type and the precise number of developers within each type. This condition enables the foundation to tailor contracts to each developer's specific type and ensure that developers select the contract accordingly. However, in practical terms, a key premise for developers' involvement in smart contract development is the assurance of nonnegative payoffs. Consequently, in this context, it is essential to apply individual rationality (IR) to guarantee the feasibility of the contract.

Definition 1 [individual rationality (IR)]: Each type-i developers gets the nonnegative payoff by choosing the contract item ω_i which is specifically designed for their type, i.e.,

$$U_D^i(\theta_i, \tau_{(i,n)}, \omega_i) \geq 0, \quad \forall i \in \mathcal{I}^\dagger. \tag{6}$$

We denote the foundation's total payoff in this scenario as $U_F^C(\tau_{\min}, \Omega)$. Thus, for the veteran developers, we can formulate the optimal multidimensional contract $\mathcal{L}_C^* = (\tau_{\min}, \Omega)$ by solving

$$\begin{aligned} &\max_{\tau_{\min},\Omega} U_F^C(\tau_{\min}, \Omega) \\ &\text{s.t. } U_D^i(\theta_i, \tau_{(i,n)}, \omega_i) \geq 0, \quad \forall i \in \mathcal{I}^\dagger. \end{aligned} \tag{7}$$

To address the contract design problem (16), we will first calculate the foundation's optimal reward $R_i^*(a_i)$ in response to any given project complexity a_i. After that, this optimal reward $R_i^*(a_i)$ will be substituted into the foundation's utility function, and we will deduce both the optimal project complexity a_i and the optimal minimum reputation value $\tau_{\min}$.

Lemma 1: For any given project complexity a_i, let the reward for type-i developers be

$$R_i^*(a_i) = g_{\text{base}} c \frac{1}{\theta_i} a_i, \quad \forall i \in \mathcal{I}^\dagger \tag{8}$$

then

1.1) $R^*(\mathbf{a})$ satisfy IR constraint

1.2) Foundation's utility is maximized if the reward for type-i developers are identical to $R^*(a_i)$

Proof: Please refer to the Appendix.

Lemma 1 demonstrates that under a complete information scenario, type-i developers are compensated by the foundation with a reward equal to their development costs, which means the foundation will design the contract that satisfies the IR constraint and offers developers zero payoff.

By substituting (8) into $\max_{\tau_{\min},\Omega} U_F^C(\tau_{\min},\Omega)$, the objective function (16) is equivalent to

$$\max_{\tau_{\min},a_i} N^\dagger(\tau_{\min}) \sum_{i=1}^{I^\dagger} \lambda_i \left[\alpha \ln(1+\theta_i a_i \psi + \tau_{\min}) - g_{\text{base}} c \frac{1}{\theta_i} a_i\right]. \tag{9}$$

From this point, we can derive the optimal project complexity for each type that maximizes the foundation's payoff. First, find the first derivative of a_i for $U_F^C(\tau_{\min},\Omega)$ in (10)

$$\frac{dU_F^C(\tau_{\min},\Omega)}{da_i} = N^\dagger(\tau_{\min})\lambda_i \left(\frac{\alpha\theta_i\psi}{1+\theta_i a_i\psi+\tau_{\min}} - g_{\text{base}} c \frac{1}{\theta_i}\right). \tag{10}$$

Next, find the second derivative of a_i for $U_F^C(\tau_{\min},\Omega)$.

$$\frac{d^2U_F^C(\tau_{\min},\Omega)}{da_i^2} = \frac{-N^\dagger(\tau_{\min})\lambda_i\alpha\theta_i^2\psi^2}{(1+\theta_i a_i\psi+\tau_{\min})^2} \tag{11}$$

It can be seen that $(d^2U_F^C(\tau_{\min},\Omega)/da_i^2) < 0$, indicating that $U_F^C(\tau_{\min},\Omega)$ is a concave function. Since the sum of concave functions remains concave, the final problem can be calculated as a concave optimization problem. We can then obtain the optimal project complexity a_i^* for type-i developer, which is

$$a_i^* = \frac{\alpha\theta_i}{g_{\text{base}} c} - \frac{1+\tau_{\min}}{\theta_i\psi}. \tag{12}$$

After substituting a_i^* back, the optimization problem is simplified as

$$\max_{\tau_{\min}} U_F^C(\tau_{\min}), \quad \forall i \in \mathcal{I}^\dagger. \tag{13}$$

Lemma 2: Define $K_C = (\sum_{i=1}^{I} \lambda_i[\alpha \ln(\alpha\psi\theta_i^2/g_{\text{base}}c) - \alpha + (g_{\text{base}}c/\theta_i^2\psi)]/\sum_{i=1}^{I}(\lambda_i g_{\text{base}}c/\theta_i^2\psi))$. Based on the contract designed for veteran developers, $U_F^C(\tau_{\min})$ is a concave function iff $\tau_{\min} \in (\tau_1,\tau_2) \cap (0,1)$ where τ_1 and τ_2 are the roots of (14). Otherwise it is an convex function of $\tau_{\min}$

$$\tau_{\min}^2 + (K_C - \mu)\tau_{\min} - (K_C\mu + 2\sigma^2). \tag{14}$$

Proof: Please refer to the appendix.

By Lemma 2, we can obtain the maxima of $U_F^C(\tau_{\min})$ within its domain. The point is to divide the domain of $U_F^C(\tau_{\min})$ into concave and convex segments respectfully. In concave segments, we can find the local maxima by using convex optimization tools, e.g., CVX. In convex segments, we then evaluate the function at the endpoints, as local maxima cannot occur in these regions. We denote the optimal strategy of $\tau_{\min}$ as $\tau_{\min}^*$ where $\tau_{\min}^* = \arg\max_{\tau_{\min}} U_F^C(\tau_{\min})$.

Theorem 1: In the complete information scenario, to maximize utility, the foundation will set $\tau_{\min} = \tau_{\min}^*$ and provide contract item ω_i^* to developer of type-i, where

$$\omega_i^* = \left(\frac{\alpha\theta_i}{g_{\text{base}}c} - \frac{1+\tau_{\min}^*}{\theta_i\psi}, \alpha - \frac{g_{\text{base}}c(1+\tau_{\min}^*)}{\theta_i^2\psi}\right).$$

Proof: Please refer to the appendix.

Theorem 1 shows that the foundation provides optimal contract item ω_i^* for each developer type and set minimum reputation value equal to $\tau_{\min}^*$ to maximize its profit. But only those developers with reputation value larger than $\tau_{\min}^*$ can get positive payoff.

B. Contract Design for Newcome Developer

The contract design for newcome developers usually entails scenarios with incomplete information. The foundation lacks prior collaboration with these developers, so it does not have detailed information about the specific type of each individual newcomer. Nonetheless, the foundation holds a broad understanding of the different types of developers and their respective proportions ($\lambda_i, \forall i \in \mathcal{I}^\dagger$) in the market. When faced with such incomplete information, the foundation can design a contract that adheres to incentive compatibility (IC) constraints, thereby ensuring that developers of each category can accept specific contract items.

Definition 2 (IC): Each type-i developer can achieve their payoff maximization by choosing the contract item ω_i specifically designed for their type, compared with any other contract ω_j $(i \neq j)$, i.e.

$$U_D^i(\theta_i, \tau_{(i,n)}, \omega_i) \geq U_D^i(\theta_i, \tau_{(i,n)}, \omega_j), \quad \forall i,j \in \mathcal{I}^\dagger, i \neq j. \tag{15}$$

We denote the foundation's total payoff in this scenario as $U_F^I(\tau_{\min},\Omega)$. Thus, for the newcome developers, we can obtain the optimal multidimensional contract $\mathcal{L}_I^* = (\tau_{\min},\Omega)$ by solving

$$\begin{aligned} &\max_{\tau_{\min},\Omega} U_F^I(\tau_{\min},\Omega) \\ &\text{s.t. } U_D^i(\theta_i,\tau_{(i,n)},\omega_i) \geq 0, \quad \forall i \in \mathcal{I}^\dagger \\ &U_D^i(\theta_i,\tau_{(i,n)},\omega_i) \geq U_D^i(\theta_i,\tau_{(i,n)},\omega_j), \quad \forall i,j \in \mathcal{I}^\dagger, i \neq j. \end{aligned} \tag{16}$$

The strategy of the optimal contract design can be obtained from the solution of (16). We first reduce the IR and IC constraints to deduce the Lemma 3 and Lemma 4.

Lemma 3: In the scenario of incomplete information, if the feasibility of the contract $\mathcal{L}_I^* = (\tau_{\min},\Omega)$ holds, the contract item for any newcome developer type in $\mathcal{I}^\dagger$ holds the following condition based on the IR constraints:

$$R(a_1) - g_{\text{base}} \frac{1}{\theta_1} c a_1 \geq 0.$$

Proof: Please refer to the appendix.

It shows that even the developers with the lowest ability θ_1 can receive nonnegative payoff if they choose contract item ω_1 which is deliberately designed for them. Furthermore, from $U_D^i(\theta_i, \tau_{(i,n)}, \omega_i) > U_D^i(\theta_i, \tau_{(i,n)}, \omega_1) \geq 0$, $\forall i \in \mathcal{I}^\dagger, i \neq 1$, we can have that for any types of developers can receive a nonnegative payoff when they select the contract ω_1. And it is notable that developers whose types are not within $\mathcal{I}^\dagger$, e.g., $\tau_{(i,n)} < \tau_{\min}$, will not be provided with any contract item, so they will not receive any development task or reward, i.e., $R(a_i) = a_i = 0, \quad \forall i \notin \mathcal{I}^\dagger$.

Lemma 4: In the scenario involved newcome developers, the feasible contracts $\mathcal{L}_I^* = (\tau_{\min}, \Omega)$ holds the following conditions based on the IC constraints.

4.1) $0 \leq R(a_1) \leq \cdots \leq R(a_{I^\dagger})$

4.2) $0 \leq a_1 \leq \cdots \leq a_{I^\dagger}$

4.3) $R(a_i) - g_{\text{base}}(c/\theta_i)a_i \geq R(a_{i+1}) - g_{\text{base}}(c/\theta_i)a_{i+1}$, $\forall i \in \{1, \ldots, I^\dagger - 1\}$

4.4) $R(a_i) - g_{\text{base}}(c/\theta_i)a_i \geq R(a_{i-1}) - g_{\text{base}}(c/\theta_i)a_{i-1}$, $\forall i \in \{2, \ldots, I^\dagger\}$.

Proof: Please refer to the appendix.

Constraints 4.1 and 4.2 show developers with lower ability will be requested to develop smart contracts with lower complexity and thus receive less reward. Constraints 4.3 and 4.4 are the local upward incentive constraints (LUIC) and local downward incentive constraints (LDIC) respectively, which characterize the relationship between type-i developers and their upward and downward neighborhoods.

Lemma 5: For any given project complexity a_i, let the reward for type-i developers be

$$R^*(a_i) = g_{\text{base}}c\left(\frac{1}{\theta_i}a_i + q_i\right)$$

where,

$$q_i = \begin{cases} 0, & \text{if } i = 1 \\ \sum_{j=1}^{i-1}\left(\frac{1}{\theta_j} - \frac{1}{\theta_{j+1}}\right)a_j, & \text{if } i = 2, \ldots, I^\dagger \end{cases} \tag{17}$$

then

5.1) $R^*(\mathbf{a})$ satisfy IR and IC constraint

5.2) Foundation's utility is maximized if the reward for type-i developers are identical to $R^*(a_i)$

Proof: Please refer to the appendix.

By substituting (17) into (16), we now have the objective function simplified as (18), shown at the bottom of the page.

From this point, we can derive the optimal project complexity for each type that maximizes the foundation's payoff based on the incomplete information scenario. We initially find the first derivative of a_i for $U_F^I(\tau_{\min}, \Omega)$, as shown in (19), and then show that (18) is a concave function with respect to a_i, such that the optimal contract can be found under each feasible $\tau_{\min}$

$$\frac{dU_F^I}{da_i} = \begin{cases} N^\dagger(\tau_{\min})\left\{\frac{\lambda_i\alpha\theta_i\psi}{1+\theta_i a_i\psi+\tau_{\min}} - g_{\text{base}}c\left[\lambda_i\frac{1}{\theta_i} + \left(1-\sum_{j=1}^{i}\lambda_j\right)\left(\frac{1}{\theta_i} - \frac{1}{\theta_{i+1}}\right)\right]\right\}, \\ \qquad\qquad \text{if } i = 1, \ldots, I^\dagger - 1 \\ N^\dagger(\tau_{\min})\left(\frac{\lambda_i\alpha\theta_i\psi}{1+\theta_i a_i\psi+\tau_{\min}} - \lambda_i g_{\text{base}}c\frac{1}{\theta_i}\right), \\ \qquad\qquad \text{if } i = I^\dagger. \end{cases} \tag{19}$$

Based on (19), the second derivative of a_i for U_F^I is as follows:

$$\frac{d^2U_F^I(\tau_{\min}, \Omega)}{da_i^2} = \frac{-N^\dagger(\tau_{\min})\lambda_i\alpha\theta_i^2\psi^2}{(1+\theta_i a_i\psi+\tau_{\min})^2} \tag{20}$$

It can be seen that $(d^2U_F^I(\tau_{\min}, \Omega)/da_i^2) < 0$, indicating that $U_F^I(\tau_{\min}, \Omega)$ is a concave function. Since the sum of concave functions remains concave, the final problem can be formulated as a concave optimization problem. We can then obtain the optimal project complexity a_i^* for type-i developer as:

$$a_i^* = \begin{cases} \frac{\lambda_i\alpha}{\lambda_i g_{\text{base}}c + (1-\sum_{j=1}^{i}\lambda_j)(1-\frac{\theta_i}{\theta_{i+1}})} - \frac{1+\tau_{\min}}{\theta_i\psi}, \\ \qquad\qquad \text{if } i = 1, \ldots, I^\dagger - 1. \\ \frac{\alpha\theta_i}{g_{\text{base}}c} - \frac{1+\tau_{\min}}{\theta_i\psi}, \quad \text{if } i = I^\dagger \end{cases} \tag{21}$$

$$\max_{\tau_{\min},\Omega} U_F^I(\tau_{\min}, \Omega) = \max_{\tau_{\min},\Omega} N^\dagger(\tau_{\min}) \sum_{i=1}^{I^\dagger-1}\left\{\lambda_i\alpha\ln(1+\theta_i a_i\psi+\tau_{\min}) - g_{\text{base}}ca_i\left[\lambda_i\frac{1}{\theta_i} + \left(1-\sum_{j=1}^{i}\lambda_j\right)\left(\frac{1}{\theta_i} - \frac{1}{\theta_{i+1}}\right)\right]\right\}$$
$$+ N^\dagger(\tau_{\min})\lambda_{I^\dagger}\left[\alpha\ln(1+\theta_{I^\dagger}a_{I^\dagger}\psi+\tau_{\min}) - g_{\text{base}}\frac{1}{\theta_{I^\dagger}}a_{I^\dagger}\right] \tag{18}$$

$$U_F^I = N^\dagger(\tau_{\min})\left(\sum_{i=1}^{I-1}\lambda_i\left(\alpha\ln\left(\frac{\lambda_i\alpha\theta_i^2\psi}{g_{\text{base}}c(\lambda_i + (1-\sum_{j=1}^{i}\lambda_j)(1-\frac{\theta_i}{\theta_{i+1}}))}\right) - \frac{\lambda_i\alpha}{\lambda_i + (1-\sum_{j=1}^{i}\lambda_j)(1-\frac{\theta_i}{\theta_{i+1}})}\right.\right.$$
$$\left.\left. + \frac{(1+\tau_{\min})g_{\text{base}}c}{\theta_i^2\psi}\right) + \lambda_I\left(\alpha\ln\left(\frac{\alpha\theta_I^2\psi}{g_{\text{base}}c}\right) - \alpha + \frac{(1+\tau_{\min})g_{\text{base}}c}{\theta_I^2\psi}\right)\right). \tag{22}$$

After substituting a_i^* back, the optimization problem is simplified to an one variable function as shown in (22), at the bottom of the previous page. For simplicity, we define constant K_I in (23), shown at the bottom of the page.

Lemma 6: In the scenario involved newcome developers, $U_F^C(\tau_{\min})$ is a concave function of $\tau_{\min}$ iff $\tau_{\min} \in (\tau_1, \tau_2) \cap (0, 1)$ where τ_1 and τ_2 are the root of (24). Otherwise it is an convex function

$$\tau_{\min}^2 + (K_I - \mu)\tau_{\min} - (K_I\mu + 2\sigma^2). \tag{24}$$

Proof: Please refer to the appendix.

We can utilize the method in Lemma 2 to obtain the maxima of $U_F^I(\tau_{\min})$ within its domain. The derived solution meets the monotonicity condition automatically when the developer type follows a uniform distribution. Otherwise, we can use the algorithm in the literature [42] to obtain the optimal solution. And here we denote the optimal strategy of $\tau_{\min}$ as $\tau_{\min}^*$ where $\tau_{\min}^* = \arg\max_{\tau_{\min}} U_F^I(\tau_{\min})$.

Theorem 2: Given newcome developers, to maximize utility, the foundation will set $\tau_{\min} = \tau_{\min}^*$ and provide contract item ω_i^* to developer of type-i as stated in (25), shown at the bottom of the page.

Theorem 2 characterizes the foundation's optimal contract designed for each newcome developer type with $\tau_{(i,n)} \geq \tau_{\min}$ in the incomplete information scenario.

VI. Performance Evaluation

In this section, we conduct numerical simulations to evaluate the performance of both the reputation filter and the contract-theoretic mechanism tailored for the ecosystem within the SCaaS market context. Unless stated otherwise, we set default parameters $\alpha = 10$, $\psi = 10$, $c = 1$, and $g_{\text{base}} = 10$. Consistent with the majority of existing studies about reputation system and contract theory, we assume that the $\tau_{(i,n)}$ is uniformly distributed in [0, 1] and N_i is uniformly distributed in [1, 10] with $I = 10$. Based on these parameters, our evaluation results are as follows.

A. Performance of the Reputation Filter

We first analyze the influence of standard deviation (std) and the mean of new projects' reputation $\tau_{(i,n)}$ on the foundation's utility. Fig. 3 illustrates that as the reputation mean increases, there is a significant increase in foundation utility, suggesting that a higher average reputation of the projects can enhance the foundation's utility. Conversely, foundation utility appears to be inversely correlated with reputation std, which indicates that higher variability in reputation correlates with lower utility. This could imply that foundations prefer a more consistent and predictable range of reputation values.

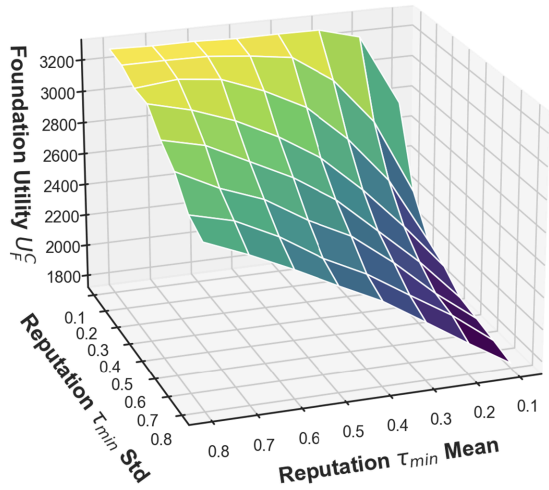
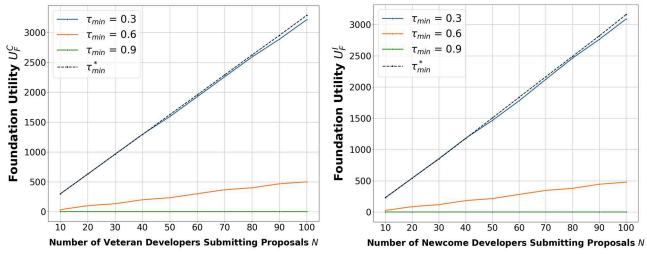


Fig. 3. Developers' utility under different contract items.

We then examine the foundation's utility across various reputation filter thresholds $\tau_{\min}$ as the number of both veteran and newcomer developers rises, depicted in Fig. 4. We observe that the optimal $\tau_{\min}$ for veteran developers is marginally higher than for newcomers. Notably, while higher project reputation values boost the foundation's profit, the optimal $\tau_{\min}$ for both groups leans towards the lower value to avoid deterring developer participation because of the high reputation demand for new projects. The utility of the foundation is markedly higher with veteran developers for all $\tau_{\min}$ levels under the same developer population, suggesting that veterans significantly enhance the foundation's utility. In both scenarios, the foundation's utility is positively correlated with an increase in the number of developers.

$$K_I = \sum_{i=1}^{I-1} \lambda_i \left(\alpha \ln \left(\frac{\lambda_i \alpha \theta_i^2 \psi}{g_{\text{base}} c (\lambda_i + (1 - \sum_{j=1}^{i} \lambda_j)(1 - \frac{\theta_i}{\theta_{i+1}}))} \right) - \frac{\lambda_i \alpha}{\lambda_i + (1 - \sum_{j=1}^{i} \lambda_j)(1 - \frac{\theta_i}{\theta_{i+1}})} + \frac{g_{\text{base}} c}{\theta_i^2 \psi} \right) + \lambda_I \left(\alpha \ln \left(\frac{\alpha \theta_I^2 \psi}{g_{\text{base}} c} \right) - \alpha + \frac{g_{\text{base}} c}{\theta_I^2 \psi} \right) \tag{23}$$

$$\omega_i^* = \begin{cases} \left(\dfrac{\lambda_i \alpha \theta_i}{g_{\text{base}} c (\lambda_i + (1 - \sum_{j=1}^{i} \lambda_j)(1 - \frac{\theta_i}{\theta_{i+1}}))} - \dfrac{1 + \tau_{\min}}{\theta_i \psi}, \dfrac{\lambda_i \alpha}{\lambda_i + (1 - \sum_{j=1}^{i} \lambda_j)(1 - \frac{\theta_i}{\theta_{i+1}})} - \dfrac{g_{\text{base}} c (1 + \tau_{\min})}{\theta_i^2 \psi} \right), \\ \qquad\qquad \text{if } i = 1, \ldots, I-1 \\ \left(\dfrac{\alpha \theta_i}{g_{\text{base}} c} - \dfrac{1 + \tau_{\min}}{\theta_i \psi}, \alpha - \dfrac{g_{\text{base}} c (1 + \tau_{\min})}{\theta_i^2 \psi} \right), \quad \text{if } i = I. \end{cases} \tag{25}$$



(a) (b)

Fig. 4. Performance of different reputation values $\tau_{\min}$. (a) Foundation's utility against veteran developers. (b) Foundation's utility against new-come developers.

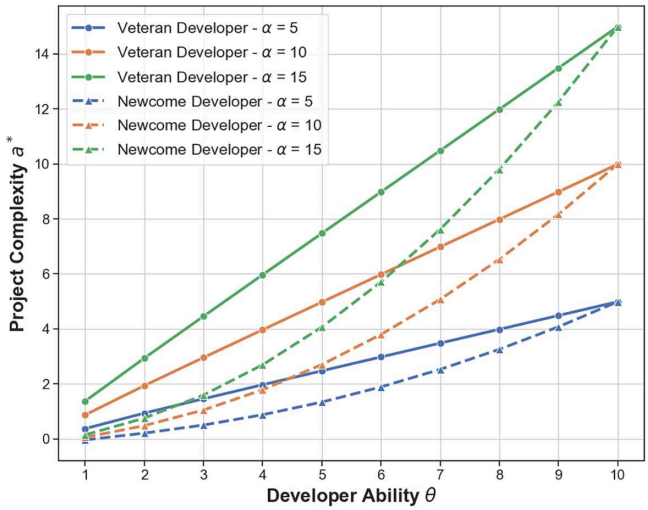


Fig. 5. Performance under satisfaction coefficient α.

B. Influence of the Satisfaction Coefficient α

Fig. 5 illustrates how the foundation's satisfaction coefficient and developers' ability affect the required project complexity. We observe that the complexity of project development for both newcome and veteran developers increases with their increasing abilities, implying constraint 4.2 holds. Notably, veteran developers start at and sustain a higher complexity level compared with the newcome developers with the same ability. In contrast, when the ability of newcome developers is at its lowest ($\theta = 1$), the complexity they manage nearly diminishes to zero. At their peak ability ($\theta = 10$), both newcome and veteran developers can handle the same project complexity. Moreover, newcome developers with greater ability experience a steeper increase in the complexity of their projects with their abilities improvement. Overall, the figure illustrates the crucial influence of veteran developers (contrasted with newcomers) in managing complex tasks. Furthermore, the alpha parameter is positively correlated with the complexity of projects across all skill levels, suggesting that a greater satisfaction (or benefit) coefficient of the foundation further motivates developers to undertake projects with higher complexity.

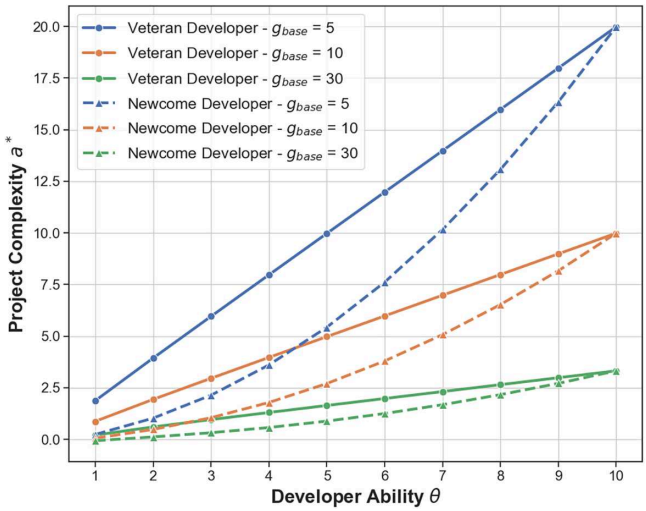


Fig. 6. Performance under basic gas price of different public blockchains g_{base}.

C. Influence of the Gas Price

Compared with Fig. 5, Fig. 6 delineates the influence of the g_{base} on developer behavior. From Fig. 6, we observe that a lower gas price motivates developers to create smart contracts with greater complexity. This is because the high gas cost characteristic serves as a deterrent to carrying out complex projects, leading developers towards the adoption of less complex projects to manage costs effectively. It is discerned from on-chain data that in EVM-based blockchains with the largest developer base, such as Polygon, BSC, and ETH, which are listed according to ascending gas fees, the number of smart contracts created last year (January 2023 to January 2024) decreased sequentially. Specifically, the creation of smart contracts on Polygon surpassed that on BSC, and BSC experienced a higher number of smart contract creations than ETH. According to [43], blockchains that support higher complexity attract developers to create more projects, which suggests that the experimental results in Fig. 4 are reflective of the actual trends witnessed.

D. Feasibility of the Proposed Optimal Contract Items

Since the results of the newcome developers and veteran developers are the same, we choose the newcome developers as an objective and compare the utility of different types of veteran developers while selecting different contract items. Fig. 7 shows that the developers' utilities are nonnegative if they select the contract items that are intended for them, which is verified by IR constraints in Lemma 3. Developers with each type result in different utilities by selecting different contract items and can get their maximum utility by selecting the contract items specially designed for their type, indicating the IC constraints in Lemma 4.

E. Influence of the Developer Type

We analyze the impact of the diversity of developer types. Fig. 8 delineates the foundation's profits within our

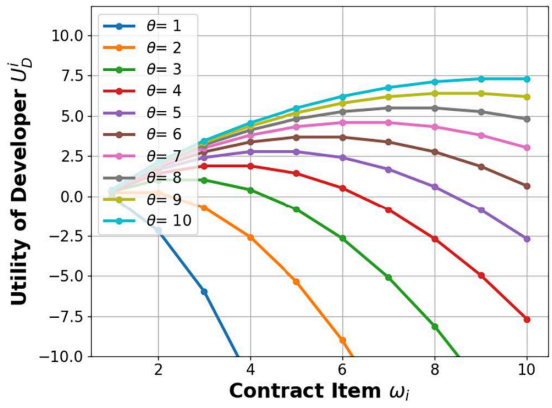


Fig. 7. Developers' utility under different contract items.

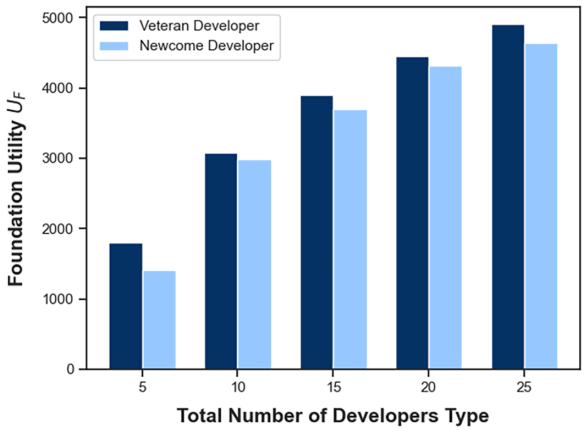


Fig. 8. Performance under different numbers of developer types.

contract-based framework. It is discernible that even when the population of developers is held constant, the publisher's profits escalate with the increasing number of developer types. This suggests that a greater variety of developer types enables the foundation to offer an expanded array of contract items, thereby attracting a broader spectrum of highly skilled developers to engage in project development. While fulfilling the demand for projects with lower complexity, it is also possible to facilitate the development of more complex projects, catering to the sophisticated requirements of users. Furthermore, it is observed that veteran developers consistently bring more utility to the foundation compared with newcome developers, with the largest difference when there are five types of developer abilities. This implies that when the number of types of developers participating in the ecosystem construction is lower, choosing veteran developers can gain higher utility for the foundation. Conversely, when the diversity of types is higher, the necessity for the foundation to stringently differentiate between a veteran and a newcomer developer diminishes.

VII. Conclusion and Future Work

In this work, we have proposed a three-stage system to help the foundation stimulate developers' involvement and reuse the existing smart contracts when developing new ones. We first introduced the SCaaS paradigm to mitigate smart contracts' redundant deployment and studied the selection and incentive mechanism of trustworthy developers in the SCaaS. Specifically, we have introduced the reputation filter to eliminate risky smart contracts based on the on-chain transaction volume. We then leveraged the multidimensional contract theory to formulate the interaction between the foundation and heterogeneous developers under newcome and veteran developers. In the incentive contracts, we have induced developers to reveal their private information by utilizing IR and IC constraints and thus optimize the foundation's utility. We also conducted experiments to validate the effectiveness of the reputation filter and the multidimensional contracts. In future work, we will consider using Soulbound Tokens [44], [45], [46] to identify the malicious newly created accounts to defraud foundation rewards, thus ensuring veteran developers' utility in our system. Simultaneously, we will collaborate with ChainIDE [47], [48], a multichain integrated development environment (IDE) that supports full-stack development, to implement and validate the proposed method. This collaboration will enable us to explore practical applications and refine our approach through real-world scenarios, which we aim to present in future work.

Appendix

A. Proof of Lemma 1

Condition 1.1: $R^*(a_i) - g_{\text{base}}c(1/\theta_i)a_i \geq 0$

Hence it satisfies IR constraint.

Condition 1.2: We prove by contradiction. Suppose there exists an optimal reward $\widetilde{R(a_i)}$ for type-i developers such that $\widetilde{R(a_i)} < R^*(a_i)$. In this case, $\widetilde{R(a_i)} - g_{\text{base}}c(1/\theta_i)a_i < 0$, which contradicts with the IR constraint. We also suppose there exists an optimal reward $\widetilde{R(a_i)}$ for type-i developers such that $\widetilde{R(a_i)} > R^*(a_i)$. In this case, $\widetilde{R(a_i)} - g_{\text{base}}c(1/\theta_i)a_i > 0$. Since the foundation can increase its utility by reducing reward until $\widetilde{R(a_i)} - g_{\text{base}}c(1/\theta_i)a_i = 0$, it contradicts with the assumption of $\widetilde{R(a_i)} > R^*(a_i)$.

Hence, by rewarding type-i developers $R^*(a_i)$, foundation's utility is maximised.

B. Proof of Lemma 2

To prove Lemma 2, we calculate the second derivative of U_F^C.

According to (12), foundation will require developer of type i to complete a smart contract of complexity $a_i^* = (\alpha\theta_i/g_{\text{base}}c) - (1 + \tau_{\min}/\theta_i\psi)$ to maximize its payoff. Therefore

$$U_F^C = N^\dagger(\tau_{\min}) \sum_{i=1}^{I} \lambda_i \left[\alpha \ln \left(1 + \theta_i \left(\frac{\alpha\theta_i}{g_{\text{base}}c} - \frac{1+\tau_{\min}}{\theta_i\psi} \right) \psi + \tau_{\min} \right) - g_{\text{base}}c \frac{1}{\theta_i} \left(\frac{\alpha\theta_i}{g_{\text{base}}c} - \frac{1+\tau_{\min}}{\theta_i\psi} \right) \right]$$

$$= N^\dagger(\tau_{\min}) \sum_{i=1}^{I} \lambda_i \left[\alpha \ln \left(\frac{\alpha\psi\theta_i^2}{g_{\text{base}}c} \right) - \alpha + \frac{(1+\tau_{\min})g_{\text{base}}c}{\theta_i^2\psi} \right].$$

For the simplicity of writing, we denote $\sum_{i=1}^{I}\lambda_i[\alpha \ln(\alpha\psi\theta_i^2/g_{\text{base}}c)-\alpha+((1+\tau_{\min})g_{\text{base}}c/\theta_i^2\psi)]$ as $h_C(\tau_{\min})$, i.e., $U_F^C=N(\tau_{\min})h_C(\tau_{\min})$.

$$\begin{cases} h_C'(\tau_{\min})=\sum_{i=1}^{I}\dfrac{\lambda_i g_{\text{base}}c}{\theta_i^2\psi} \\ h_C''(\tau_{\min})=0. \end{cases} \tag{26}$$

Under the assumption that $\tau_{(i,n)}\sim N(\mu,\sigma^2)$, the density function of $\tau_{(i,n)}$ is

$$f(\tau_{\min})=\frac{1}{\sqrt{2\pi}\sigma}e^{\frac{-(\tau_{\min}-\mu)^2}{2\sigma^2}}. \tag{27}$$

And the cumulative density function of $\tau_{(i,n)}$ is

$$F(\tau_{\min})=Pr(\tau_{(i,n)}\le\tau_{\min})=\int_{-\infty}^{\frac{\tau_{\min}-\mu}{\sigma}}\frac{1}{\sqrt{2\pi}}e^{\frac{-x^2}{2}}dx. \tag{28}$$

Let $\mathbb{1}_{\tau_{(i,n)}\ge\tau_{\min}}$ be an indicator function such that:

$$\mathbb{1}_{\tau_{(i,n)}\ge\tau_{\min}}=\begin{cases}1, \tau_{(i,n)}\ge\tau_{\min}\\ 0, \tau_{(i,n)}<\tau_{\min}.\end{cases} \tag{29}$$

Since $\tau_{(i,n)}$ is normal distributed, we can obtain the discrete density function of $\mathbb{1}_{\tau_{(i,n)}\ge\tau_{\min}}$

$$\begin{aligned} &Pr(\mathbb{1}_{\tau_{(i,n)}\ge\tau_{\min}}=1)=Pr(\tau_{(i,n)}\ge\tau_{\min})=1-F(\tau_{\min})\\ &Pr(\mathbb{1}_{\tau_{(i,n)}\ge\tau_{\min}}=0)=Pr(\tau_{(i,n)}<\tau_{\min})=F(\tau_{\min}). \end{aligned} \tag{30}$$

Then N_{pass}, the exact number of developers that pass the reputation filter, can be derived by

$$N_{\text{pass}}=\sum_{n=1}^{N}\mathbb{1}_{\tau_{(i,n)}\ge\tau_{\min}}. \tag{31}$$

N_{pass} follows a binomial distribution with population N and probability $1-F(\tau_{\min})$

$$N_{\text{pass}}\sim b(N,1-F(\tau_{\min})). \tag{32}$$

Therefore, $N(\tau_{\min})$, the number of developers passing the reputation filter approximated by the foundation, is given by the expectation of N_{pass}

$$\begin{aligned} N(\tau_{\min})&=E[N_{\text{pass}}]=N(1-F(\tau_{\min}))\\ &=N\int_{-\infty}^{\frac{\mu-\tau_{\min}}{\sigma}}\frac{1}{\sqrt{2\pi}}e^{\frac{-x^2}{2}}dx. \end{aligned} \tag{33}$$

Then differentiate $N(\tau_{\min})$ twice with respect to $\tau_{\min}$

$$\begin{cases} N'(\tau_{\min})=\dfrac{-N}{\sigma\sqrt{2\pi}}e^{\frac{-(\mu-\tau_{\min})^2}{2\sigma^2}}<0\\ N''(\tau_{\min})=\dfrac{-N(\mu-\tau_{\min})}{\sigma^3\sqrt{2\pi}}e^{\frac{-(\mu-\tau_{\min})^2}{2\sigma^2}}<0. \end{cases} \tag{34}$$

Then differentiate U_F^C twice with respect to $\tau_{\min}$

$$\begin{cases} U_F^{C'}=N'(\tau_{\min})h_C(\tau_{\min})+N(\tau_{\min})h_C'(\tau_{\min})\\ U_F^{C''}=2N'(\tau_{\min})h_C'(\tau_{\min})+N''(\tau_{\min})h_C(\tau_{\min}). \end{cases} \tag{35}$$

By computation, $U_F^{C''}<0$ iff $\tau_{\min}^2+(K_C-\mu)\tau_{\min}-(K_C\mu+2\sigma^2)<0$. Recall the domain of $U_F^C(\tau_{\min})$ is $(0,1)$. Therefore, U_F^C is a concave function of $\tau_{\min}$ iff $\tau_{\min}\in(\tau_1,\tau_2)\cap(0,1)$ where τ_1 and τ_2 are the roots of (14). Proof of Lemma 2 is completed.

C. Proof of Theorem 1

According to Lemma 2, the final problem is a concave optimization problem and we can solve for a value of $\tau_{\min}$ such that U_F^C is maximized, denoted as $\tau_{\min}^{\#}$.

Therefore, optimal minimum reputation value is $\tau_{\min}^*=\tau_{\min}^{\#}$ and, according to (12), the optimal project complexity required from type-i developer is

$$a_i^*=\frac{\alpha\theta_i}{g_{\text{base}}c}-\frac{1+\tau_{\min}^*}{\theta_i\psi}. \tag{36}$$

Subsequently, the reward given to developers belong to type-i once they pass the filter is

$$\begin{aligned} R(a_i^*)&=g_{\text{base}}c\frac{1}{\theta_i}\left(\frac{\alpha\theta_i}{g_{\text{base}}c}-\frac{1+\tau_{\min}^*}{\theta_i\psi}\right)\\ &=\alpha-\frac{g_{\text{base}}c(1+\tau_{\min}^*)}{\theta_i^2\psi}. \end{aligned} \tag{37}$$

The optimal contract item provided to developer type-i is

$$\phi_i^*=\left(\frac{\alpha\theta_i}{g_{\text{base}}c}-\frac{1+\tau_{\min}^*}{\theta_i\psi},\alpha-\frac{g_{\text{base}}c(1+\tau_{\min}^*)}{\theta_i^2\psi}\right). \tag{38}$$

D. Proof of Lemma 3

According to IR constraint

$$R(a_i)-g_{\text{base}}\frac{1}{\theta_i}ca_i\ge 0,\forall i\in\{1,\dots,I^\dagger\} \tag{39}$$

we have

$$R(a_1)-g_{\text{base}}\frac{1}{\theta_1}ca_1\ge 0. \tag{40}$$

Proof of Lemma 3 is completed.

E. Proof of Lemma 4

To prove conditions 4.1 and 4.2, first we prove that, under IC constraint, $\theta_i<\theta_j$ if and only if $a_i\le a_j$.

We prove by contradiction. Suppose that there exists the case where $\theta_i<\theta_j$ and $a_i>a_j$. Then we have

$$(\theta_j-\theta_i)(a_j-a_i)<0. \tag{41}$$

According to the IC constraints, we have

$$\begin{cases} R(a_i)-g_{\text{base}}c\dfrac{1}{\theta_i}a_i\ge R(a_j)-g_{\text{base}}c\dfrac{1}{\theta_i}a_j\\ R(a_j)-g_{\text{base}}c\dfrac{1}{\theta_j}a_j\ge R(a_i)-g_{\text{base}}c\dfrac{1}{\theta_j}a_i. \end{cases} \tag{42}$$

Combining the above two equations, we have

$$\begin{aligned} &-\frac{1}{\theta_i}a_i-\frac{1}{\theta_j}a_j\geq-\frac{1}{\theta_i}a_j-\frac{1}{\theta_j}a_i\\ &\Leftrightarrow\left(\frac{1}{\theta_j}-\frac{1}{\theta_i}\right)(a_j-a_i)\leq 0\\ &\Leftrightarrow(\theta_j-\theta_i)(a_j-a_i)\geq 0 \end{aligned} \tag{43}$$

which is in contradiction with (41). Hence, under IC constraint, $\theta_i<\theta_j$ if and only if $a_i\leq a_j$.

Condition 4.2: We have proven $\theta_i<\theta_j$ if and only if $a_i\leq a_j$, and since $\theta_1\leq\cdots\leq\theta_{I^\dagger}$, we have $0\leq a_1\leq\cdots\leq a_{I^\dagger}$.

Condition 4.1: For type-i developers, the following IC constraints must be satisfied:

$$\begin{aligned} &R(a_i)-g_{\text{base}}c\frac{1}{\theta_i}a_i\geq R(a_j)-g_{\text{base}}c\frac{1}{\theta_i}a_j\\ &\Leftrightarrow\frac{1}{\theta_i}(a_j-a_i)\geq R(a_j)-R(a_i). \end{aligned} \tag{44}$$

Thus, we have if $a_i>a_j$, then $0>(1/\theta_i)(a_j-a_i)\geq R(a_j)-R(a_i)$, which means that $R(a_i)>R(a_j)$. Since $0\leq a_1\leq\cdots\leq a_{I^\dagger}$, we have $0\leq R(a_1)\leq\cdots\leq R(a_{I^\dagger})$

Condition 4.3: According to IC constraint, we have

$$R(a_i)-g_{\text{base}}c\frac{1}{\theta_i}a_i\geq R(a_j)-g_{\text{base}}c\frac{1}{\theta_i}a_j,\quad\forall i,j\in\{1,2,\ldots,I^\dagger\}. \tag{45}$$

Here, substitute j with $j=i+1$, and then we have

$$R(a_i)-g_{\text{base}}c\frac{1}{\theta_i}a_i\geq R(a_{i+1})-g_{\text{base}}c\frac{1}{\theta_i}a_{i+1},\quad\forall i\in\{1,\ldots,I^\dagger-1\}. \tag{46}$$

Condition 4.4: Substitute j with $j=i-1$ in (45), and then we have

$$R(a_i)-g_{\text{base}}c\frac{1}{\theta_i}a_i\geq R(a_{i-1})-g_{\text{base}}c\frac{1}{\theta_i}a_{i-1},\quad\forall i\in\{2,\ldots,I^\dagger\}. \tag{47}$$

Proof of Lemma 4 is completed.

F. Proof of Lemma 5

Condition 5.1: We first show that $R^*(\mathbf{a})$ satisfied IR constraint

$$R^*(a_i)-g_{\text{base}}c\frac{1}{\theta_i}a_i=g_{\text{base}}cq_i\geq 0,\forall i\in\{1,\ldots,I^\dagger\}. \tag{48}$$

Therefore, $R^*(\mathbf{a})$ satisfies IR constraint.

We then show that $R^*(\mathbf{a})$ satisfied IC constraint through induction. We denote by $\Omega(k)$ the subset of Ω, which contains the first k contract items in Ω, i.e., $\Omega(k)=\{(a_i,R(a_i))|i=1,\ldots,k\}$.

We first verify that $\Omega(1)$ satisfies IC constraint. Since there is only one contract item, the IC constraint must be satisfied.

Then, we show that if $\Omega(k)$ satisfies IC constraint, then $\Omega(k+1)$ also satisfies IC constraint. This corresponds to two aspects:

1) For the new type $(k+1)$, the IC constraint is satisfied, i.e.,

$$R(a_{k+1})-g_{\text{base}}c\frac{1}{\theta_{k+1}}a_{k+1}\geq R(a_i)-g_{\text{base}}c\frac{1}{\theta_{k+1}}a_i,\quad\forall i\in\{1,\ldots,k\}. \tag{49}$$

2) For existing types $1,\ldots,k$, the IC constraint is still satisfied in the presence of type $k+1$, i.e.,

$$R(a_i)-g_{\text{base}}c\frac{1}{\theta_i}a_i\geq R(a_j)-g_{\text{base}}c\frac{1}{\theta_i}a_j\quad\forall i\in\{1,\ldots,k+1\}. \tag{50}$$

We first prove (49). Since $\Omega(k)$ satisfy IC constraint, the IC constraint for type-k developers holds, i.e.,

$$R(a_i)-g_{\text{base}}c\frac{1}{\theta_k}a_i\leq R(a_k)-g_{\text{base}}c\frac{1}{\theta_k}a_k,\quad\forall i\in\{1,\ldots,k\}. \tag{51}$$

Besides, according to condition 4.4, we have

$$R(a_{k+1})\geq R(a_k)+g_{\text{base}}c\frac{1}{\theta_{k+1}}(a_{k+1}-a_k). \tag{52}$$

By combining (51) and (52), we have

$$\begin{aligned} &R(a_{k+1})+R(a_k)-g_{\text{base}}c\frac{1}{\theta_k}a_k\\ &\geq R(a_k)+g_{\text{base}}c\frac{1}{\theta_{k+1}}(a_{k+1}-a_k)+R(a_i)-g_{\text{base}}c\frac{1}{\theta_k}a_i \end{aligned} \tag{53}$$

which can be written as

$$\begin{aligned} &R(a_{k+1})-g_{\text{base}}c\frac{1}{\theta_{k+1}}a_{k+1}\\ &\geq R(a_i)-g_{\text{base}}c\frac{1}{\theta_{k+1}}a_k+g_{\text{base}}c\frac{1}{\theta_k}(a_k-a_i)\\ &\geq R(a_i)-g_{\text{base}}c\frac{1}{\theta_{k+1}}a_k\\ &\geq R(a_i)-g_{\text{base}}c\frac{1}{\theta_{k+1}}a_i,\quad\forall i\in\{1,\ldots,k\}. \end{aligned} \tag{54}$$

The last two inequalities are both base on that the condition 4.2, i.e., $a_i\leq a_k,\forall i\in\{1,\ldots,k\}$. Thus, the IC constraint for type-(k + 1) developers hold

Then, we prove (50). It is obvious that proving (50) is equivalent to proving the following:

$$R(a_i)-g_{\text{base}}c\frac{1}{\theta_i}a_i\geq R(a_{k+1})-g_{\text{base}}c\frac{1}{\theta_i}a_{k+1},\quad\forall i\in\{1,\ldots,k\}. \tag{55}$$

According to condition 4.3, we have

$$R(a_{k+1})\leq R(a_k)+g_{\text{base}}c\frac{1}{\theta_k}(a_{k+1}-a_k). \tag{56}$$

$$h_I(\tau_{\min}) = \sum_{i=1}^{I-1} \lambda_i \left(\alpha \ln \left(\frac{\lambda_i \alpha \theta_i^2 \psi}{g_{\text{base}} c (\lambda_i + (1 - \sum_{j=1}^{i} \lambda_j)(1 - \frac{\theta_i}{\theta_{i+1}}))} \right) - \frac{\lambda_i \alpha}{(\lambda_i + (1 - \sum_{j=1}^{i} \lambda_j)(1 - \frac{\theta_i}{\theta_{i+1}}))} + \frac{(1+\tau_{\min}) g_{\text{base}} c}{\theta_i^2 \psi} \right) + \lambda_I \left(\alpha \ln \left(\frac{\alpha \theta_I^2 \psi}{g_{\text{base}} c} \right) - \alpha + \frac{(1+\tau_{\min}) g_{\text{base}} c}{\theta_I^2 \psi} \right). \tag{63}$$

Also, since IC constraint for type-i ($i \in \{1, \ldots, k\}$) developers hold, i.e.,

$$R(a_i) - g_{\text{base}} c \frac{1}{\theta_i} a_i \geq R(a_k) - g_{\text{base}} c \frac{1}{\theta_i} a_k. \tag{57}$$

By combining the above two inequality (56) and (57), we have

$$\begin{aligned} &R(a_{k+1}) + R(a_k) - g_{\text{base}} c \frac{1}{\theta_i} a_k \\ &\leq R(a_k) + g_{\text{base}} c \frac{1}{\theta_k} (a_{k+1} - a_k) + R(a_i) - g_{\text{base}} c \frac{1}{\theta_i} a_i \end{aligned} \tag{58}$$

which can be written as

$$\begin{aligned} &R(a_i) - g_{\text{base}} c \frac{1}{\theta_i} a_i \\ &\geq R(a_{k+1}) - g_{\text{base}} c \frac{1}{\theta_i} a_k - g_{\text{base}} c \frac{1}{\theta_k} (a_{k+1} - a_k) \\ &\geq R(a_{k+1}) - g_{\text{base}} c \frac{1}{\theta_i} a_k - g_{\text{base}} c \frac{1}{\theta_i} (a_{k+1} - a_k) \\ &= R(a_{k+1}) - g_{\text{base}} c \frac{1}{\theta_i} a_{k+1}, \quad \forall i \in \{1, \ldots, k\}. \end{aligned} \tag{59}$$

Hence, $R^*(\mathbf{a})$ satisfies IR and IC constraints.

Condition 5.2: We then show that the reward in (17) gives the maximum utility to the foundation. Assume there is a reward $\widetilde{R(\mathbf{a})}$ such that, given fixed project complexity set $\mathbf{a}$, $N^\dagger(\tau_{\min}) \sum_{i=1}^{I^\dagger} \lambda_i \widetilde{R(a_i)} < N^\dagger(\tau_{\min}) \sum_{i=1}^{I^\dagger} \lambda_i R^*(a_i)$. Thus, there is at least one $\widetilde{R(a_i)} < R^*(a_i)$.

To make the contract feasible, based on condition 4.4, the following constraint on $\widetilde{R(a_i)}$ have to be satisfied:

$$\widetilde{R(a_i)} \geq \widetilde{R(a_{i-1})} + g_{\text{base}} c \frac{1}{\theta_i} (a_i - a_{i-1}) \tag{60}$$

which can be written as

$$\begin{aligned} \widetilde{R(a_{i-1})} &\leq \widetilde{R(a_i)} + g_{\text{base}} c \frac{1}{\theta_i} (a_{i-1} - a_i) \\ &< R^*(a_i) + g_{\text{base}} c \frac{1}{\theta_i} (a_{i-1} - a_i) = R^*(a_{i-1}). \end{aligned} \tag{61}$$

Continue the above process, we can obtain that

$$\widetilde{R(a_1)} < R^*(a_1) = g_{\text{base}} c \frac{1}{\theta_1} a_1 \tag{62}$$

which violates IR constraint of type-1 developers.

Therefore, the foundation's utility function is maximised by the reward in (17).

G. Proof of Lemma 6

To prove Lemma 6, we calculate the second derivative of U_F^I.

For the simplicity of writing, we define $h_I(\tau_{\min})$ in (63), shown at the top of the page. In this case, $U_F^I = N(\tau_{\min}) h_I(\tau_{\min})$

$$\begin{cases} h_I'(\tau_{\min}) = \sum_{i=1}^{I} \dfrac{\lambda_i g_{\text{base}} c}{\theta_i^2 \psi} \\ h_I''(\tau_{\min}) = 0. \end{cases} \tag{64}$$

Then differentiate U_F^I twice with respect to $\tau_{\min}$

$$\begin{cases} U_F^{I'} = N'(\tau_{\min}) h_I(\tau_{\min}) + N(\tau_{\min}) h_I'(\tau_{\min}) \\ U_F^{I''} = 2N'(\tau_{\min}) h_I'(\tau_{\min}) + N''(\tau_{\min}) h_I(\tau_{\min}) < 0. \end{cases} \tag{65}$$

By computation, $U_I^{I''} < 0$ iff $\tau_{\min}^2 + (K_I - \mu)\tau_{\min} - (K_I \mu + 2\sigma^2) < 0$. Recall the domain of $U_F^I(\tau_{\min})$ is $(0, 1)$.Therefore, U_F^I is a concave function of $\tau_{\min}$ iff $\tau_{\min} \in (\tau_1, \tau_2) \cap (0, 1)$ where τ_1 and τ_2 are the roots of (24). Proof of Lemma 5 is completed.

H. Proof of Theorem 2

According to Lemma 5, the final problem is a cnoncave optimization problem and we can solve for a value of $\tau_{\min}$ such that U_F^I is maximized, denoted as $\tau_{\min}^{\#}$.

Therefore, optimal minimum reputation value is $\tau_{\min}^* = \tau_{\min}^{\#}$ and the optimal project complexity required from type-i developer is a_I^* in (21). Subsequently, the reward given to developers belong to type-i once they pass the filter is

$$R(a_i^*) = g_{\text{base}} c \frac{1}{\theta_i} a_i^*. \tag{66}$$

The optimal contract item provided to developer type-i is

$$\phi_i^* = \left(a_I^*, g_{\text{base}} c \frac{1}{\theta_i} a_i^* \right). \tag{67}$$

REFERENCES

[1] H. Taherdoost, "Smart contracts in blockchain technology: A critical review," *Information*, vol. 14, no. 2, p. 117, 2023.

[2] M. Nofer, P. Gomber, O. Hinz, and D. Schiereck, "Blockchain," *Bus. Inf. Syst. Eng.*, vol. 59, pp. 183–187, Mar. 2017.

[3] N. Kannengießer, M. Pfister, M. Greulich, S. Lins, and A. Sunyaev, *Bridges Between Islands: Cross-Chain Technology for Distributed Ledger Technology*, pp. 5298–5307, 2020.

[4] M. Goint, C. Bertelle, and C. Duvallet, "Secure access control to data in off-chain storage in blockchain-based consent systems," *Mathematics*, vol. 11, no. 7, p. 1592, 2023.

[5] S. N. Khan, F. Loukil, C. Ghedira-Guegan, E. Benkhelifa, and A. Bani-Hani, "Blockchain smart contracts: Applications, challenges, and future trends," *Peer-to-peer Netw. Appl.*, vol. 14, pp. 2901–2925, Apr. 2021.

[6] G. Yu et al., "Toward web3 applications: Easing the access and transition," *IEEE Trans. Comput. Social Syst.*, vol. 11, no. 5, pp. 6098–6111, Oct. 2024.

[7] N. Six, N. Herbaut, and C. Salinesi, "Blockchain software patterns for the design of decentralized applications: A systematic literature review," *Blockchain: Res. Appl.*, vol. 3, no. 2, 2022, Art. no. 100061.

[8] T. Sharma, Z. Zhou, A. Miller, and Y. Wang, "Exploring security practices of smart contract developers," 2022, *arXiv:2204.11193*.

[9] G. A. Oliva, A. E. Hassan, and Z. M. Jiang, "An exploratory study of smart contracts in the ethereum blockchain platform," *Empirical Softw. Eng.*, vol. 25, pp. 1864–1904, Mar. 2020.

[10] F. A. Alaba, H. A. Sulaimon, M. I. Marisa, and O. Najeem, "Smart contracts security application and challenges: A review," *Cloud Comput. Data Sci.*, vol. 5, no. 1, pp. 15–41, 2024.

[11] V. Bracamonte and H. Okada, "An exploratory study on the influence of guidelines on crowdfunding projects in the ethereum blockchain platform," in *Proc. Social Inform.: 9th Int. Conf., SocInfo*, Oxford, UK: Springer, Sep. 2017, pp. 347–354.

[12] M. B. Neitz, "The influencers: Facebook's libra, public blockchains, and the ethical considerations of centralization," *NCJL Tech.*, vol. 21, p. 41, Dec. 2019.

[13] S. Baltes and S. Diehl, "Towards a theory of software development expertise," in *Proc. 2018 26th ACM joint Meeting Eur. Softw. Eng. Conf. Symp. Foundations Softw. Eng.*, 2018, pp. 187–200.

[14] N. Archak, "Money, glory and cheap talk: analyzing strategic behavior of contestants in simultaneous crowdsourcing contests on topcoder. com," in *Proc. 19th Int. Conf. World Wide Web*, 2010, pp. 21–30.

[15] M. Daltayanni, *Reputation Systems in Labor and Advertising Marketplaces,* Santa Cruz, California, USA: University of California, 2015.

[16] D. Yang, Y. Ji, Z. Kou, X. Zhong, and S. Zhang, "Asynchronous federated learning with incentive mechanism based on contract theory," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC),* Piscataway, NJ, USA: IEEE Press, 2024, pp. 1–6.

[17] H. Ma, Y. Gu, H. Wu, L. Xing, and X. Zhang, "A dual incentive mechanism based on graph attention neural network and contract in mobile opportunistic networks," *Phys. Commun.*, vol. 67, 2024, Art. no. 102485.

[18] N. Phumchusri and P. Amornvetchayakul, "Machine learning models for predicting customer churn: a case study in a software-as-a-service inventory management company," *Int. J. Bus. Intell. Data Mining*, vol. 24, no. 1, pp. 74–106, 2024.

[19] E. Keller and J. Rexford, "The "platform as a service" model for networking," *INM/WREN*, vol. 10, pp. 95–108, Apr. 2010.

[20] S. Bhardwaj, L. Jain, and S. Jain, "Cloud computing: A study of infrastructure as a service (IaaS)," *Int. J. Eng. Inf. Technol.*, vol. 2, no. 1, pp. 60–63, 2010.

[21] M. N. Moeti, "Infrastructure as a service adoption model for south african universities using thematic analysis," *South Afr. J. Inf. Manage.*, vol. 26, no. 1, p. 14, 2024.

[22] M. Samaniego, U. Jamsrandorj, and R. Deters, "Blockchain as a service for IoT," in *Proc. IEEE Int. Conf. Internet Things (iThings) IEEE Green Comput. Commun. (GreenCom) IEEE Cyber, Phys. Social Comput. (CPSCom) IEEE Smart Data (SmartData),* Piscataway, NJ, USA: IEEE Press, 2016, pp. 433–436.

[23] B. U. I. Khan et al., "Blockchain-enhanced sensor-as-a-service (SEaaS) in IoT: Leveraging blockchain for efficient and secure sensing data transactions," *Information*, vol. 15, no. 4, p. 212, 2024.

[24] A. M. Tripathi and L. S. Umrao, "Integrated fuzzy decision tree based blockchain federated safety-as-a-service for IIoT," *Cluster Comput.*, vol. 28, no. 2, p. 95, 2024.

[25] W.-W. Li, W. Meng, K.-H. Yeh, and S.-C. Cha, "Trusting computing as a service for blockchain applications," *IEEE Internet Things J.*, vol. 10, no. 13, pp. 11326–11342, 2023.

[26] H. Duan, J. Li, S. Fan, Z. Lin, X. Wu, and W. Cai, "Metaverse for social good: A university campus prototype," in *Proc. 29th ACM Int. Conf. Multimedia*, 2021, pp. 153–161.

[27] S. Fan, H. Zhang, Y. Zeng, and W. Cai, "Hybrid blockchain-based resource trading system for federated learning in edge computing," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2252–2264, Feb. 2021.

[28] S. Fan, J. Zhao, R. Zhao, Z. Wang, and W. Cai, "CryptoArcade: A cloud gaming system with blockchain-based token economy," *IEEE Trans. Cloud Comput.*, vol. 11, no. 3, pp. 2445–2458, Jul./Sep. 2023.

[29] J. Park, S. Jeong, and K. Yeom, "Smart contract broker: Improving smart contract reusability in a blockchain environment," *Sensors*, vol. 23, no. 13, p. 6149, 2023.

[30] F. Khan, I. David, D. Varro, and S. McIntosh, "Code cloning in smart contracts on the ethereum platform: An extended replication study," *IEEE Trans. Softw. Eng.*, vol. 49, no. 4, pp. 2006–2019, Apr. 2022.

[31] X. Shen, W. Li, H. Xu, X. Wang, and Z. Wang, "A reuse-oriented visual smart contract code generator for efficient development of complex multi-party interaction scenarios," *Appl. Sci.*, vol. 13, no. 14, p. 8094, 2023.

[32] Y. A. Hsain, N. Laaz, and S. Mbarki, "SCEditor: A graphical editor prototype for smart contract design and development." *Int. J. Adv. Comput. Sci. Appl.*, vol. 15, no. 3, pp. 1185–1195, 2024.

[33] S. Fu, X. Huang, L. Liu, and Y. Luo, "BFCRI: A blockchain-based framework for crowdsourcing with reputation and incentive," *IEEE Trans. Cloud Comput.*, vol. 11, no. 2, pp. 2158–2174, Apr./Jun. 2023.

[34] H. Jiao, J. Liu, J. Li, and C. Liu, "A framework for reputation bootstrapping based on reputation utility and game theories," in *Proc. IEEE 10th Int. Conf. Trust, Secur. Privacy Comput. Commun.,* Piscataway, NJ, USA: IEEE Press, 2011, pp. 344–351.

[35] E. Parhizkar, M. H. Nikravan, R. C. Holte, and S. Zilles, "Combining direct trust and indirect trust in multi-agent systems," in *Proc. IJCAI*, 2020, pp. 311–317.

[36] H. Bangui, M. Ge, and B. Buhnova, "Deep-learning based reputation model for indirect trust management," *Procedia Comput. Sci.*, vol. 220, pp. 405–412, Jan. 2023.

[37] M. Dai, Z. Su, Y. Wang, and Q. Xu, "Contract theory based incentive scheme for mobile crowd sensing networks," in *Proc. Int. Conf. Sel. Topics Mobile Wireless Netw. (MoWNeT),* Piscataway, NJ, USA: IEEE Press, 2018, pp. 1–5.

[38] Z. Su, L. Liu, M. Li, X. Fan, and Y. Zhou, "ServiceTrust: Trust management in service provision networks," in *Proc. IEEE Int. Conf. Services Comput.*, Piscataway, NJ, USA: IEEE Press, 2013, pp. 272–279.

[39] M. Zhou and A. Mockus, "Developer fluency: Achieving true mastery in software projects," in *Proc. 18th ACM SIGSOFT Int. Symp. Found. Softw. Eng.*, 2010, pp. 137–146.

[40] B. Wang et al., "Secrets of RLHF in large language models part II: Reward modeling," 2024, *arXiv:2401.06080*.

[41] N. Ding, Z. Fang, and J. Huang, "Optimal contract design for efficient federated learning with multi-dimensional private information," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 1, pp. 186–200, Jan. 2021.

[42] Z. Xiong, J. Kang, D. Niyato, P. Wang, H. V. Poor, and S. Xie, "A multi-dimensional contract approach for data rewarding in mobile networks," *IEEE Trans. Wireless Commun.*, vol. 19, no. 9, pp. 5779–5793, Sep. 2020.

[43] A. Vacca, A. Di Sorbo, C. A. Visaggio, and G. Canfora, "A systematic literature review of blockchain and smart contract development: Techniques, tools, and open challenges," *J. Syst. Softw.*, vol. 174, 2021, Art. no. 110891.

[44] T. V. Tumati, "SBTCert: A soulbound token certificate verification system," Ph.D. dissertation, California State Univ., Northridge, LA, USA, 2023.

[45] T. J. Chaffer and J. Goldston, "On the existential basis of self-sovereign identity and soulbound tokens: An examination of the "self" in the age of web3," *J. Strategic Innov. Sustainability*, vol. 17, no. 3, pp. 1–9, 2022.

[46] T. V. Tumati, Y. Tian, and X. Jiang, "A soulbound token certificate verification system (sbtcert): Design and implementation," in *Proc. IEEE 14th Annu. Comput. Commun. Workshop Conf. (CCWC),* Piscataway, NJ, USA: IEEE Press, 2024, pp. 0345–0350.

[47] H. Qiu, X. Wu, S. Zhang, V. C. Leung, and W. Cai, "ChainIDE: A cloud-based integrated development environment for cross-blockchain smart contracts," in *Proc. IEEE Int. Conf. Cloud Comput. Technol. Sci. (CloudCom),* Piscataway, NJ, USA: IEEE Press, 2019, pp. 317–319.

[48] W. Liang, Y. Liu, C. Yang, S. Xie, K. Li, and W. Susilo, "On identity, transaction, and smart contract privacy on permissioned and permissionless blockchain: A comprehensive survey," *ACM Comput. Surveys*, vol. 56, no. 12, pp. 1–35, 2024.

Jinghan Sun received the M.Eng. degree in mechanical engineering from Huazhong University of Science and Technology, Wuhan, China. She is working toward the Ph.D. degree in computer and information engineering with the School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen, China.

Currently, she works as a Research Assistant with Mohamed bin Zayed University of Artificial Intelligence, Abu Dhabi, UAE. Her research interests include blockchain, token economy, and Web3.

Hou-Wan Long is working toward the B.Sc. degree in risk management science from The Chinese University of Hong Kong, Hong Kong SAR.

His research interests include blockchain, token economy, and financial technology.

Hong Kang received the B.Eng. degree in electronic information engineering from The University of Electronic Science and Technology of China, Chengdu, China, and University of Glasgow, Scotland, U.K., in 2021. He is currently working toward the M.Phil. degree in computer and information engineering with the School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen, China.

He is working as a Research Assistant with the Human-Crypto Society Laboratory, Shenzhen, China. His research interests include blockchain, mechanism design, and edge intelligence.

Zhixuan Fang (Member, IEEE) received the B.S. degree in physics from Peking University, Beijing, China, in 2013, and the Ph.D. degree in computer science from Tsinghua University, Beijing, China, in 2018.

Currently, he is a tenure-track Assistant Professor with the Institute for Interdisciplinary Information Sciences (IIIS), Tsinghua University. His research interests include the design and analysis of multiagent systems, blockchain, and networked systems.

Abdulmotaleb El Saddik (Fellow, IEEE) is a Distinguished Professor and is an internationally recognized scholar who has made seminal contributions to the knowledge and understanding of multimedia computing, communications and applications. His visionary work looks toward the establishment of Digital Twins using AI, Haptics, AR/VR/Haptics and 5G that allow people to interact in real-time with one another as well as with their digital representation in the Metaverse. He has been extremely productive of high-quality research and impact. He is the Editor in Chief of the *ACM Transactions on Multimedia Computing, Communications and Applications (ACM TOMM)*, a Senior Associate Editor of IEEE CONSUMER ELECTRONICS MAGAZINE (IEEE MCE), and Guest Editor for several Transactions and Journals. He has authored and co-authored ten books and more than 600 peer-reviewed articles and five patents and chaired more than 50 conferences and workshops. He has received research grants and contracts totaling more than $30 M. He has supervised more than 160 researchers. He is the author of the book *Haptics Technologies: Bringing Touch to Multimedia*.

Dr. El Saddik is a fellow of the Royal Society of Canada, IEEE, the Canadian Academy of Engineering and the Engineering Institute of Canada. He is an ACM Distinguished Scientist and has received several awards, including the Friedrich Wilhelm Bessel Award from the German Humboldt Foundation, the IEEE Instrumentation and Measurement Society Technical Achievement Award. He also received IEEE Canada C.C. Gotlieb (Computer) Medal and A.G.L. McNaughton Gold Medal for important contributions to the field of computer engineering and science and the IEEE TCSC Achievement Award for Excellence in Scalable Computing.

Wei Cai (Senior Member, IEEE) received the B.Eng. degree in software engineering from Xiamen University, Xiamen, China, in 2008, the M.S. degree in electrical engineering and computer science from Seoul National University, Seoul, Korea, in 2011, and the Ph.D. degree in electrical and computer engineering from The University of British Columbia (UBC), Vancouver, Canada, in 2016.

From 2016 to 2018, he was a Postdoctoral Research Fellow with UBC. Currently, he is an Assistant Professor of Computer Engineering with the School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen, China. He is serving as the Director of the Human-Crypto Society Laboratory, Shenzhen, China, as well as the Director of the CUHK(SZ)-White Matrix Joint Metaverse Laboratory, Shenzhen, China. He has co-authored more than 100 journal and conference papers in the areas of distributed/decentralized systems.

Dr. Cai is now serving as an Associate Editor for IEEE TRANSACTIONS ON COMPUTATIONAL SOCIAL SYSTEMS (TCSS), IEEE TRANSACTIONS ON CLOUD COMPUTING (TCC), *ACM Transactions on Multimedia Computing, Communications and Applications (TOMM)*, program Co-Chair for ACM Workshop on Network and Operating Systems Support for Digital Audio and Video in 2023 and open-source software competition Co-Chair for ACM Multimedia in 2023. He was a recipient of six Best Paper Awards. He is a member of the ACM.