УДК 004.738.5 + 005.8 http://doi.org/10.15407/pp2024.02-03.351

L. Katerynych, M. Veres, K. Riabov

TRANSFORMING GOVERNANCE: ENABLING SCALABLE AND ADAPTABLE DECENTRALIZED NETWORKS ON EVM-COMPATIBLE BLOCKCHAIN

This article explores scalable and adaptable governance through decentralized networks, enabling collective decision-making and evolution without necessitating a complete system overhaul when original functionalities become obsolete. It delves into the utilization of interconnected Smart Contracts on the EVM-based blockchain to resolve foundational governance issues. However, the principal advantages are compromised if the system requires continual redeployment to adapt to environmental changes.

We introduce a solution that combines a role-based access system and a modular system contracts architecture to enhance the system's scalability and adaptability. This approach allows for modifications and scaling at any time through community proposals and voting, catering to the specific needs of its members and eliminating the need for manual configuration by a centralized entity for new modules or functionalities. Members can propose configuration parameters for a new module, and with community majority approval, the system can adapt and scale, safeguarding the interests of its members, provided there is cooperation within the majority of the community.

Key words Decentralized Autonomous Organization, Decentralized Governance, WEB3, Role-Based Access System, Ethereum, Blockchain, Solidity

Л.О. Катеринич, М.М. Верес, К.С. Рябов

ТРАНСФОРМАЦІЯ УПРАВЛІННЯ: СТВОРЕННЯ МАСШТАБОВАНИХ ТА АДАПТОВАНИХ ДЕЦЕНТРАЛІЗОВАНИХ МЕРЕЖ НА ОСНОВІ EVM-СУМІСНОГО БЛОКЧЕЙНУ

Ця стаття досліджує масштабоване та адаптивне управління через децентралізовані мережі, що уможливлює колективне прийняття рішень та еволюцію без необхідності в повному перебудовані системи, коли первинні функціональні можливості стають застарілими. Розглядається використання взаємопов'язаних Смарт-контрактів на блокчейні, базованому на EVM для вирішення фундаментальних проблем управління. Проте, основні переваги компрометуються, якщо система потребує постійного перевидання для адаптації до змін у середовищі.

Ми представляємо рішення, яке поєднує систему доступу на основі ролей та архітектуру системи модульних контрактів для підвищення масштабованості та адаптивності системи. Цей підхід дозволяє вносити зміни та масштабувати систему у будь-який час через спільні пропозиції та голосування спільноти. Водночас існує можливість врахування конкретних потреб учасників і уникнення необхідності вручну конфігурувати нові модулі або функціональні можливості централізованою сутністю. Учасники можуть пропонувати параметри конфігурації для нового модуля і за підтримки більшості у спільноті адаптувати систему та здійснювати масштабування, захищаючи інтереси своїх учасників за умови співпраці більшості у спільноті.

Ключові слова: Decentralized Autonomous Organization, Decentralized Governance, WEB3, Role-Based Access System, Ethereum, Blockchain, Solidity.

.

1. Introduction

Navigating decision-making on the Internet poses a significant challenge in establishing a robust system where members cannot manipulate outcomes and only eligible parties can participate. On another side, it was important to prevent 'double-spending' problem, so users cannot reply the same action again and again in order to get a benefit. The solution to

the problem was proposed by the adoption of the Bitcoin [1]. However, it did not allow creating a complex governance structures with custom rules that cannot be omitted.

The adaptation of Smart Contracts [2, 3] on the Ethereum network has addressed this issue, operating under the principle of 'code is law' [4]. While this approach serves smaller communities and projects focusing on straightforward financial management effectively, it struggles with continuously evolving systems that need to adapt to real-world trends and changes.

Ethereum, inherently a perpetually evolving protocol, enables the development of a diverse ecosystem of commercial products, allowing community interaction and participation in decision-making processes. A prevalent instance is the creation of platforms, known as Decentralized Autonomous Organizations (DAOs), where community members or investors can influence the project's trajectory in the WEB3 sphere. Given the irreversible nature of user actions and Smart Contracts in Ethereum, significant modifications to the contract logic are challenging, and manual upgrades through a Proxy Upgrade pattern [5] can be risky. Contracts also face a size limitation, hindering scalability when the available space is exhausted. Additionally, the inability to adapt to new products and protocols in the Ethereum ecosystem can render older DAOs obsolete.

To address these limitations, a modular architecture incorporating a role-based access system is essential, allowing seamless integration with new protocols and functionalities without redeploying the entire system. This approach not only eliminates the need for system redeployment and reconfiguration but also empowers the community to modify the system's functionality through voting, ensuring security and sustainability as long as the majority actively cooperate.

2. Role-Based Access System

This article introduces a Role-Based Access System (RBAS) serving as a pivotal connector between system components, establishing a set of rules that delineate access to system resources. These rules are designed to be transparent and comprehensible to community members, offering enhanced flexibility to the system.

Where each community member or the system itself can be considered an Entity that performs specific Actions on resources.

In the prevalent Ethereum ecosystem, most contracts depend on account addresses to determine resource access, a method that restricts the system to specific addresses. This limitation becomes problematic in systems with numerous contracts, confining them to precise implementations that cannot be easily and securely modified.

RBAS, in contrast, abstracts resource access to parties with specific permissions, granted through community voting, simplifying the management of system component relationships. It centralizes all access rules, unlike traditional approaches where access to protected functions requires extensive contract code analysis. Despite the additional gas usage, RBAS compensates by facilitating a more manageable and adaptable system.

By leveraging RBAS, which grants permissions through community voting, management of system components is simplified and adaptability is enhanced. For example, in Corporate Governance [6], implementing RBAS in DAOs illustrates how it can supplement traditional business structures, enabling dynamic, decentralized regulatory solutions and showcasing a compliant approach to managing corporate entities.

The core capability of the RBAS is its ability to assimilate external modules while maintaining tight control over them and ensuring that the integrated components conform to the control protocols established in the system. This integration is key to extending and diversifying the functionality of the system, allowing the inclusion of various components such as DAO bridges, Uniswap, Treasuries and other elements that require the influence of DAO management.

In practical terms, this means that any component, once integrated, automatically adapts to the rules and protocols of system management, inheriting established norms and operational frameworks. This integration is critical for consistency and uniformity across the system, ensuring that all components, regardless of their origin and nature, operate

within a common governance structure, reducing the risks associated with inconsistencies and non-compliance.

In essence, integrating external modules using RBAS not only enriches the ecosystem with diverse functionality, but also strengthens the governance structure by ensuring uniformity and compliance across all components. This is an example of a balanced synergy between extension and governance, allowing the system to evolve and adapt while maintaining its underlying principles and integrity..

2.1. Roles, Resources, Entities, Actions, and Permissions

Our proposed solution initiates with the delineation of roles, resources, entities, actions, and permissions within the system, as illustrated in Fig. 1.

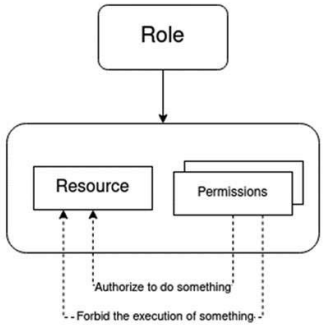

Fig. 1. Role, Resource, Permissions Model.

Entity (E): An individual or component that interacts with the system.

$$E = \{e_1, e_2, \dots, e_n\}$$

Action (A): An operation performed by an entity on a resource.

$$A = \{a_1, a_2, \dots, a_n\}$$

Permission ($Perm$): A set of rules determining access to the system's resources, either allowing or denying such access.

$$Perm = \{accept, reject\}$$

Resources (Res): A set of functions grouped either as a Smart Contract ($\mathcal{SC}$) or as functions within an SC.

$$Res = \{\mathcal{SC}_1, \mathcal{SC}_2, \dots, \mathcal{SC}_n\}$$

Role ($Role$): An amalgamation of one resource and one or more permissions, structuring relationships within the system in a simple yet effective manner.

$$Role = Res \times Perm$$

To enhance efficiency and optimization, we used the concept of a group abstraction.

Group (Grp): A compilation of members sharing identical roles.

$$Grp = \{g_1, g_2, \dots, g_k\}$$

Instead of granting roles directly to members ($\mathcal{M}$), they are role assigned to groups.

$$\forall g \in Grp, \exists Role_g \subseteq Role$$

$$\mathcal{M} \in Grp \Rightarrow \mathcal{M}\ has\ Role(Grp)$$

Consequently, a member's inclusion in a group bestows upon them all the permissions allocated to that group.

Essentially, a group operates as an array to which a specific role can be granted. Consequently, each member within this array inherits the permissions associated with the group, allowing for a streamlined allocation of roles and permissions, as illustrated at Fig. 2.

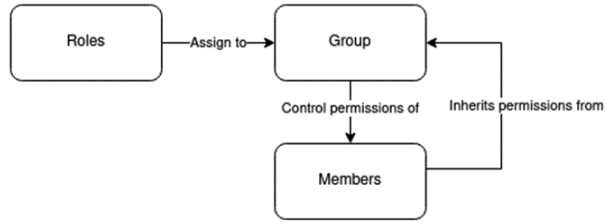

Fig. 2. Group-Based Role Allocation.

This structured approach sets the stage for defining the core properties and formal guarantees of the RBAS system.

2.2. Core Properties and Formal Guarantees

The systematization above not only streamlines internal relationships but also ensures a structured approach to resource and permission allocation. This allows us to extract the following properties:

Role Assignment: Each entity $e \in E$ is assigned one or more roles, which collectively

determine the actions they are authorized to perform on resources:

$$\forall e \in E, \exists Role_e \subseteq Role$$

Permission Enforcement ($Perform$): For an entity e to perform an action a on a resource r, the entity must possess a role that includes the corresponding permission:

$$Perform(e, a, r) \Rightarrow \exists Role_e \ni (r, a)$$

Access Control Function ($Access$): An access control function $Access$ verifies whether an entity e has the necessary permissions to perform an action on a resource r:

$$Access = \begin{cases} true, if\ \exists Role_e \ni (r, a) \\ false, \quad otherwise \end{cases}$$

By combining the core properties and system promytives, we are achieving the following formal garanties:

Correctness and Consistency: The RBAC system ensures that only authorized actions are performed, preventing unauthorized access and modifications and ensuring that role assignments and permissions are uniformly enforced across all actions:

$$\forall (e, a, r)\ if\ Access(e, a, r) = true$$
$$then\ Perform(e, a, r)$$

The RBAC system, as formalized above, provides a robust framework for managing access and permissions within the decentralized network. It ensures that every action is authorized, maintaining the integrity and security of the system while allowing for scalability and adaptability. The subsequent sections will elucidate its application within the context of a DAO.

3. Modular Smart Contracts Architecture

To construct a system capable of scaling indefinitely, a modular System Contract architecture is imperative, aligning with the standards proposed by Nick Mudge in ERC-2535 [7]. Given the restrictive 24KB [4] contract size limitation, this architecture organizes a collection of Smart Contracts under the ERC-2535 standard to accommodate extensive systems.

Each system component is a distinct module or contract, enabling the DAO to integrate with a myriad of protocols within the Ethereum ecosystem and incorporate new functionalities with static addresses. This modularity negates the need for users to navigate through multiple System Contracts to locate specific functionalities provided by individual Smart Contracts within the system.

The DAO is essentially defined as:

$$DAO = \{Storage, \{mod_1, mod_2, \dots, mod_p\}\}$$

Where the $Storage$ represents the state of the DAO and a $mod_i, i \in \{1 \dots p\}$ is a distinct component or contract within the system.

The adaptability inherent in Modular Smart Contracts Architecture allows DAOs to seamlessly integrate or modify modules, ensuring continuous compliance with evolving disclosure regulations and aligning with the transparency and consumer protection needs of the decentralized finance ecosystem [8].

As visualized in Fig. 3, the actual data is housed in the main entity, which, in this instance, is a DAO. This main entity uses specific delegate calls to interact with and utilize the functionalities provided by separate entities, referred to as facets.

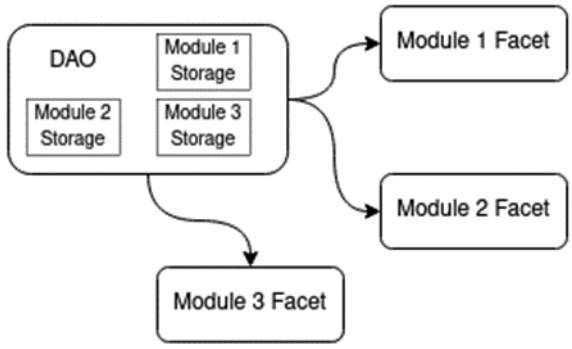

Fig. 3. Modular Smart Contracts Architecture Diagram.

4. Governance Structure

Traditional governance modules predominantly rely on the voting power of community members, represented through various means such as the quantity of ERC-20 [9], NFT [10], ERC-1155 [11] tokens or native

currency locked within the system. For a proposal to gain acceptance, it must achieve a requisite quorum and a majority of votes.

However, as governance expands with increasing user participation, the likelihood of suboptimal decisions escalates. To mitigate this, we introduce an Expert Group, comprised of members with the authority to veto specific community-selected proposals and to initiate expert-specific proposals, necessitating an in-depth understanding of the system.

Formally Expert Group ($ExpGrp$) is a set of entities with special permissions:

$$ExpGrp = \{e_1, e_2, \dots, e_k\} \; where \; e_i \in E$$

This additional protective layer aims to reduce the risk of system stagnation by ensuring that decisions are meticulously scrutinized and are reflective of informed and expert opinions, thereby enhancing the robustness and reliability of the governance structure.

At every level of the governance structure, from the creation to the execution of proposals, the community diligently works to uphold the security and stability of the system, as illustrated in Fig. 4. This visualization depicts the community's commitment to preventing any disruptions and ensuring the continuous, smooth operation of the system.

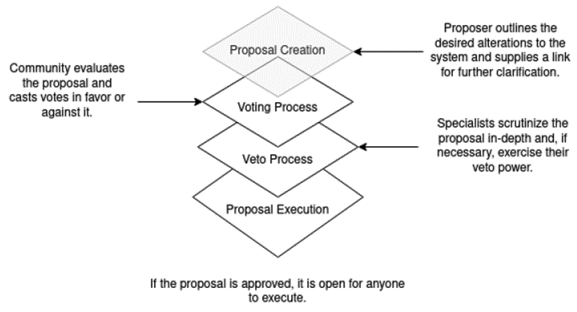

Fig. 4. Layers of Governance.

Employing a multi-layer strategy is a prudent governance practice adopted by various protocols within the Ethereum ecosystem to bolster system security [12, 13].

4.1. Insights into the Voting Mechanism

The voting process (VP), in its most comprehensive configuration, is bifurcated into two pivotal phases: voting and vetoing. The uniqueness of this process is attributed to the concept of the 'voting situation,' a set of parameters defining the target module of the proposal and specifying the entities endowed with the authority to veto the proposal.

4.1.1. The Role of Veto in Governance

The veto process is integral to the governance of the DAO, acting as a protective mechanism to halt proposals that may diverge from the DAO's foundational principles and constitution. This process is reserved for Experts, appointed during the DAO governance process, and serves as a safeguard to ensure the alignment of all proposals with the DAO's values and objectives.

Experts during the voting phase, assess the proposals based on their adherence to the DAO. If a proposal is deemed detrimental or misaligned with the DAO's interests, they can exercise their veto power to prevent its implementation, thereby preserving the integrity and values of the organization. The veto right can be described as following:

$$VetoRight(VP, e) = \begin{cases} 1, if \; e \in ExpGrp \\ 0, \quad otherwise \end{cases}$$

4.1.2. Voting and Veto Configuration

The voting process within a DAO is a fundamental mechanism allowing Members or Experts to participate in decision-making by voting on various proposals. This process is highly configurable, allowing for modifications to proposal types and associated settings, such as quorum and majority, through the parameter voting.

Voting power (V_p) is determined by the number of locked tokens in the DAO.

$$V_p(e) = \text{tokens locked by } e$$

Therefore,

$$VoteRight(VP, e) = \begin{cases} true, V_p(e) > 0 \\ false, otherwise \end{cases}$$

To ensure a proposal's validity, it must meet a specified quorum (Q), which is the minimum number of votes required.

$$Q = \sum_{e \in E} V_p(e) \times q_e \quad where\, q_e \in [0,1]$$

Additionally, it should not be vetoed by more than n experts in the ExpGrp. The veto is true if more than n experts decide to veto the proposal:

$$Veto(VP, e) = \begin{cases} true, if \sum_{e \in ExpGrp} VetoRight(VP, e) > n \\ false, \quad otherwise \end{cases}$$

The system offers three types of voting: Partially Restricted, Restricted, and Non-Restricted Voting, each serving different purposes and allowing various levels of participation from experts and the community.

When a proposal is submitted, entities cast their votes. The validation of a proposal depends on meeting the quorum and veto criteria. If the condition is satisfied, the proposal passes and is executed; otherwise, it fails.

$$Execute(VP) = true, \text{if}$$
$$\sum_{e \in E} V_p(e) \geq Q \; and \; Veto(VP) \neq true$$
$$else \; false$$

Ultimately, due to its modular architecture, any default functionality can be expanded, exemplifying the system's scalability and adaptability. For instance, features such as the ability to retract a vote can be seamlessly integrated.

5. DAO Architecture

This article explores a governance system designed to adapt and scale within an ever-evolving environment, referred to as a DAO within the Ethereum community. For effective DAO operation, the integration of key components such as Permission Manager, Vault, Voting, Member Storage, and Parameters Storage is essential. The integration of these components, as outlined in Fig. 5, forms the fundamental structure of the DAO, enabling optimal performance and efficient scalability while maintaining core functionality.

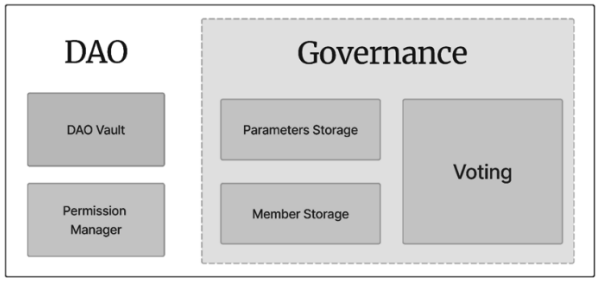

Fig. 5. Core DAO Framework.

These components will be discussed in detail in the following subchapters.

5.1. Permission Manager

The Permission Manager Module is a foundational component of the DAO, grounded in RBAS theory, and serves as the nucleus for recording and storing permissions and groups. It is pivotal for integrations and upgrades, acting as the gateway for all interactions with the DAO, ensuring a structured and secure interface. Efficiency is crucial, as every segment of the system interacts with this module, with the modular architecture and unified shared storage significantly reducing costs and lowering gas usage when accessing internal storage.

5.2. Vault

The Vault component is the entry point to the DAO Governance System, where users deposit tokens to engage in governance activities. It secures tokens during voting periods and releases them post-voting to prevent governance attacks like double-voting or vote manipulation. By delegating the locking mechanism to the Vault, the system supports a wide range of tokens, enhancing versatility and inclusivity. Its modular architecture allows for the easy integration of new tokens and functionalities, such as ERC-5484 [14], ensuring a secure and reliable voting environment and reinforcing the overall governance structure.

5.3. Voting

The Voting component orchestrates the voting process, allowing community members to initiate proposals and cast votes while

enabling Experts to exercise veto power. It is intricately linked to the Permission Manager to validate eligibility for initiating votes, casting votes, and vetoing. The component uses a generic interface to configure key voting parameters such as quorum, majority, and duration, managing both community and expert proposals. It interacts with Member Storage for veto functions and Parameter Storage to adjust voting parameters. By managing permissions through RBAS, the voting module adapts to new functionalities and ensures structured, secure governance, making it the cornerstone of the system's governance activities.

5.4. Member and Parameter Storages

The Member and Parameter Storage modules are crucial for the DAO, optimizing efficiency for components like the Voting and Permission Manager. Member Storage organizes and manages Experts, acting as a whitelist for community-elected members. Parameter Storage manages DAO parameters, simplifying the adjustment and monitoring of system settings. The modular architecture allows for the seamless integration of new functionalities post-deployment, enhancing the governance structure through RBAS. This integration reduces complexity, improves auditability and transparency, and contributes to a coherent and manageable governance system.

5.5. Module Integration Flow

Combining the different modules results in a robust system that can be easily scaled up. Thus, including the AirDrop module requires one proposal from the community, as shown in Fig. 6.

Due to the modular smart contract architecture, the new AirDrop module can be easily integrated into the DAO core, allowing access to the entire DAO storage. In addition, the RBAS ensures that this module is simultaneously integrated into the DAO governance system. This means that any configuration desired by the DAO can be achieved, such as a configuration where only experts manage the new module. However, the initial adoption of such rules is subject to community approval.

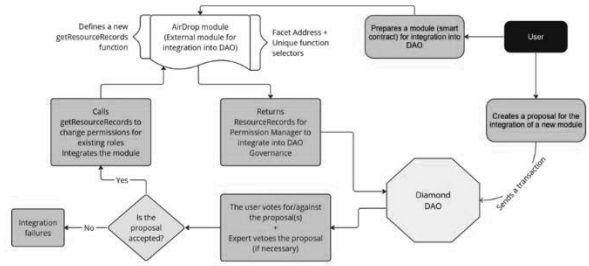

Fig. 6. Example of Practical Module Integration

5.6. System Properties

Sumarizing all the components describe above; our system gains two importan guarantees:

Liveness: As soon as the system is updated with new modules, it remains actual and relevant. Formally, for any module ($\mathrm{mod_i}$), if ($\mathrm{mod_i}$) is integrated at time (t), then the system state (S) reflects the integration at ($\mathrm{t} + \epsilon$):

$$\forall mod_i, \exists t \text{ such that integrated } (mod_i, t) \Rightarrow S(t+\epsilon) = \text{ updated}$$

Safety: With a guarantee from the voting process and the RBAC system, the system stays secure, ensuring that only authorized and secure actions are allowed. Formally, for any action a on resource r by entity e, if the action is performed, then e has the necessary permissions:

$$\forall e \in E, \forall a \in A, \forall r \in R, Perform(e,a,r) \Rightarrow Access(e,a,r) = true$$

These properties ensure that the DAO remains functional and secure, adapting to changes while maintaining strict access controls.

6. Conclusion

This article has presented a scalable and adaptable governance system on the Ethereum network, designed to circumvent the inherent limitations of the Ethereum protocol. The foundation of this system is the RBAS, which orchestrates the interactions between various components within a DAO. However, RBAS alone is insufficient to overcome the constraints related to Smart Contract size and

the immutable nature of contract logic post-deployment inherent in the Ethereum protocol.

To address these challenges, we introduced a modular system architecture, allowing the DAO to expand and integrate a diverse range of modules while maintaining coherent governance. This modular approach, coupled with the Expert governance structure, enhances the security and reliability of the system, ensuring robust protection against potential vulnerabilities.

We have also delineated a core set of components essential for initiating the functionality of the DAO, laying the groundwork for a system that is not only scalable and adaptable but also secure and governed with precision and transparency.

References

1. F. P. Miller, A. F. Vandome, Information Theory, Alpha Press, 2009. Nakamoto, Satoshi, Bitcoin: A Peer-to-Peer Electronic Cash System, 2008. URL: https://ssrn.com/abstract=3440802 or http://dx.doi.org/10.2139/ssrn.3440802
2. Ethereum Foundation, Introduction to smart contracts, 2023. URL: https://ethereum.org/en/smart-contracts/.
3. A.M. Antonopoulos, G. Wood, Mastering ethereum: building smart contracts and dapps, O'Reilly Media, 2018. URL: https://github.com/ethereumbook/ethereumbook/.
4. Berlin version 2bcdb2d, Ethereum: a secure decentralised generalised transaction ledger, Ethereum & Parity, 2023. URL: https://ethereum.github.io/yellowpaper/paper.pdf.
5. OpenZeppelin, Proxy upgrade pattern. URL: https://docs.openzeppelin.com/upgrades-plugins/1.x/proxies.
6. (Minneapolis), W. A. K., Professor at University of Saint Thomas School of Law. (2021). Blockchain-Based Corporate Governance. Stanford Journal of Blockchain Law & Policy. Retrieved from https://stanford-jblp.pubpub.org/pub/blockchain-corporate-governance
7. EIP-2535, Diamonds, Multi-Facet Proxy, Nick Mudge, 2020. URL: https://eips.ethereum.org/EIPS/eip-2535.
8. Center, C. B., Agnes N. Williams Professor of Law at Georgetown University Law. (2022). Disclosure, Dapps and DeFi. Stanford Journal of Blockchain Law & Policy. Retrieved from https://stanford-jblp.pubpub.org/pub/disclosure-dapps-defi
9. EIP-20, Token standard, Ethereum Foundation, 2015. URL: https://eips.ethereum.org/EIPS/eip-20.
10. EIP-721, Non-Fungible token standard, Ethereum Foundation, 2018. URL: https://eips.ethereum.org/EIPS/eip-721.
11. EIP-1155, Multi token standard, Ethereum Foundation, 2018. URL: https://eips.ethereum.org/EIPS/eip-1155.
12. Synthetix, Governance, 2023. URL: https://synthetix.io/governance.
13. Q Dvelopment AG, Q whitepaper, v1.0, Q Foundation, 2018. URL: https://q.org/files/Q_Whitepaper_v1.0.pdf.
14. EIP-5484, Consensual soulbound tokens, Ethereum Foundation, 2022. URL: https://eips.ethereum.org/EIPS/eip-5484.

Одержано: 10.04.2024
Внутрішня рецензія отримана: 19.02.2024
Зовнішня рецензія отримана: 08.03.2024

Про авторів:

[1]*Катеринич Лариса*,
кандидат фізико-математичних наук,
доцент
http://orcid.org/0000-0001-7837-764X .

[1]*Верес Максим*,
кандидат фізико-математичних наук,
доцент.
http://orcid.org/0000-0002-8512-5560.

[1]*Рябов Кирило*,
студент.
https://orcid.org/0009-0003-4118-8492.

Місце роботи авторів:

[1] Taras Shevchenko National University of Kyiv
тел. +38-044- 259-05-11
E-mail: katerynych@gmail.com
https://csc.knu.ua/uk/department/iss