Efficient Token Transfer Using ERC-20 Decentralized Exchange

Mettupalle Chinnaiahgari Venkata ViswasReddy
Department of Computer Engineering and Technology
Chaitanya Bharathi Institute of Technology
Hyderabad, India
viswasmc238@gmail.com

Vootkuri Sai Charan Reddy
Department of Computer Engineering and Technology
Chaitanya Bharathi Institute of Technology
Hyderabad, India
vootkurisaicharan@gmail.com

Gudeme Jaya Rao
Department of Computer Engineering and Technology
Chaitanya Bharathi Institute of Technology
Hyderabad, India
jayaraog_cet@cbit.ac.in

N.Rama Devi
Department of Computer Engineering and Technology
Chaitanya Bharathi Institute of Technology
Hyderabad, India
nramadevi_cet@cbit.ac.in

2024 International Conference on Intelligent Systems for Cybersecurity (ISCS) | 979-8-3503-7523-7/24/$31.00 ©2024 IEEE | DOI: 10.1109/ISCS61804.2024.10581224

***Abstract*—The implementation of blockchain-based token transfers has emerged as a transformative tool for innovation and enhanced financial inclusion. To implement Blockchain based token transfers, smart contracts play a pivotal role in automating complex processes, enhancing efficiency, and driving innovation for tokenized assets and decentralized applications (dApps) that can be run on Ethereum. With rising popularity of smart contracts in 2015, necessity for a standardized token approach within the Ethereum ecosystem is recognized. To address interoperability issues and challenges in exchanging tokens between different applications, the Ethereum Request for Commet-20 (ERC20) standard was introduced through the Ethereum Request for Comment (ERC) process. Since then, several numerous schemes have been proposed efforts to enhance token transfers and implementations while adhering to ERC-20 standards. Despite the substantial progress made in optimizing the functionality and efficiency of token transfers, achieving a truly user-friendly experience has remained an ongoing challenge. To address this challenge, several schemes have been proposed to refine the user interface and experience associated with ERC-20 token transfers. Our research work proposes a practical approach to address concerns by creating a simple user-friendly decentralized web application (Web3) using smart contracts that adhere to ERC-20 standards. By focusing on the development of a seamless user experience, the system developed aims to empower users to send tokens effortlessly by connecting to various wallets available in the market.**

***Index Terms*—ERC20 Tokens, dApps, web3, Ethereum Request for Comment-20 Standards, Decentralization.**

I. Introduction

In recent years, blockchain technology has witnessed unprecedented growth, revolutionizing various industries with its decentralized and transparent nature. One of the pivotal advancements within the blockchain ecosystem is the development of ERC-20 tokens on the Ethereum platform [1].ERC20 was created by Ethereum developers on behalf of the broader Ethereum community in 2015 and was officially adopted in September 2017. In previous years, to create a standard of this type for Ethereum, a developer or group of developers submitted what was known as an Ethereum Improvement Proposal (EIP), describing the new functionality along with its specific protocols and standards. A committee then reviewed, approved, amended and finalized that EIP—at that point, it became an ERC. These tokens adhere to a standardized set of rules, enabling seamless integration and interoperability across a myriad of decentralized applications (DApps) and exchanges.

The Ethereum Request for Comments (ERC) 20 standard has emerged as the foundation for countless tokenized projects, fostering innovation and expanding the possibilities of decentralized finance (DeFi) and token economies.ERC-20 tokens embody a set of protocol standards outlining a specific set of rules and functionalities that Ethereum-based tokens must adhere to. These tokens exhibit a level of fungibility [2], meaning they are interchangeable on a one-to-one basis, fostering a sense of uniformity within the Ethereum ecosystem. Smart contracts, the self-executing contracts with coded terms of agreement, underpin ERC-20 tokens, providing a secure and transparent framework for their creation and operation. It allows developers to create smart-contract-enabled tokens that can be used with other products and services.

The ERC20 standard has played a crucial role in the rise of Initial Coin Offering (ICOs) and token crowdfunding. ERC-20 tokens are popular because they are widely supported by major cryptocurrency exchanges and wallets. This broad support makes ERC-20 tokens easy to buy, sell, and access, attracting users, investors, and traders. The straightforward process of listing and trading ERC-20 tokens on different platforms adds to their popularity in the broader cryptocurrency community.

In response to these challenges, the developed system aims to develop a user-friendly decentralized web application (web3) utilizing smart contracts compliant with ERC-20 standards [3]. The primary goal is to empower users to seamlessly send tokens by connecting to various available wallets in the market, thereby enhancing accessibility and usability within the blockchain ecosystem.

TABLE I
COMPREHENSIVE STUDY

S.No	*Title of paper*	*Techniques used*	*Research gap*
1	Token-Based Access Control .	access control tokens, TBAC, security, trustworthiness	The paper does not discuss the long-term sustainability of TBAC in the face of evolving blockchain technologies and changing security landscapes. Research on maintaining the integrity and relevance of TBAC over time is valuable
2	PROPOSAL OF USER-FRIENDLY DESIGN OF NFT MARKETPLACE	Evaluation of three public blockchain platforms (Fantom, Avalanche, and Polygon) for minting and transferring NFTs.	The research primarily focuses on technical aspects. However, understanding user experience challenges and adoption barriers for NFT marketplaces is essential to foster their widespread acceptance. Future research could explore these non-technical dimensions
3	Towards Saving Blockchain Fees via Secure and Cost-Effective Batching of Smart-Contract Invocations	Batching smart-contract invocations	it introduces the IBATCH middleware system for secure smart-contract invocation batching on Ethereum and evaluates its cost-effectiveness. However, it could benefit from further exploration of potential security vulnerabilities or limitations of the IBATCH system
4	Proposal of a smart contract-based security token management system	Tokenization, Secure Tokens Management	While tokenization has unlocked various possibilities for asset representation, including real estate, it has not fully addressed the concerns of investors regarding the secure transfer of assets
5	Blockchain Based Direct Benefit Transfer System For Subsidy Delivery	Distributed Ledger, Cryptographic Hash, Direct Benefit Transfer (DBT), Digital Identity	Challenges include scalability and dependence on Aadhaar ID. Future work suggests a new distributed architecture for improved data integrity and privacy

II. LITERATURE REVIEW

Many authors have proposed various techniques and schemes in regarding Digital assets transfer by using blockchain technology and cryptocurrencies. P.Shamili et.al, [4] explores how Ethereum and its ERC20 tokens, is changing finance. One important part discussed is Uniswap, a protocol allowing people to trade ERC20 tokens without a central authority. However, it points out some problems, especially with ERC20 token transfers.More research has to be done to make ERC20 tokens work better in DeFi applications.Yibo Wang et.al, [5] extends works done by [4] by introducing IBATCH, a middleware system aimed at securely and efficiently batching smart-contract invocations on the Ethereum network.Also they have discussed on cost saving analysis of IBATCH to optimize blockchain costs for existing DApps on Ethereum.

The concern on improving how security tokens are managed using smart contracts and blockchain technology are discussed by Shingo Fujimoto,et.al, [6]. They have worked out on the limitations of current security token offerings (STOs) and suggests ways to make them work better in real-world businesses and explains the different stages they go through, like creating tokens, selling them, managing them afterward, and trading them.But they have not discussed clear use of ERC-20 standards in STOs.Where as GUOHUA GAN,et.al, [7] focuses on Token-Based Access Control (TBAC) integrated with blockchain and smart contracts, by emphasizing the significance and application of Access Control Tokens (ACTs) in various scenarios such as medical examinations, transportation, education, and intellectual property rights. The system may face difficulties in ensuring consistent and reliable token transfer, particularly in high-concurrency scenarios. Therefore, further enhancements may be needed to address these limitations and ensure secure and trustworthy token transfer within traditional distributed systems.

Sandip Ranjan Behera, et.al, [8] identifies practical solutions to the outlined issues, including the addition of widely used payment methods, enhanced platform security, improved user interface, responsive UI design, and comprehensive tutorials and customer support by studying the user interactions and experiences.In the context of token transfer,it may require additional considerations to ensure secure and private token transfers within the marketplace.Apart from these, Joshua Priest.et.al [9] contextualizes the study within existing research on Ethereum blockchain and smart contract development.They have given overview of ERC Ethereum's scalability challenges. Authors established a foundation for understanding ERC-20 standards token development also highlighting the broader challenges and opportunities in the Ethereum ecosystem. Moreover, the authors have not given broader idea of connecting the smart contracts with available wallets for Fungible token transfers.

Sayed Azain Jaffer,et.al, [10] proposed a blockchain-based solution to address issues such as data privacy [11]–[13], misuse of subsidy amounts, empowerment of citizens,.The paper outlines a systematic research methodology, presenting a detailed system architecture utilizing Hyperledger Fabric and Composer for blockchain infrastructure [14]. The results highlight the potential benefits of the proposed system while acknowledging limitations related to deployment and dependence on Aadhaar ID. Moreover, the authors had given a lot of scope in using tokens for securing the data.Sutida Narongsak,et.al, [15] introduces a blockchain-based solution to address the prevalent issue of counterfeit academic credentials such as QR code authentication and blockchain-based certificate verification systems.However, one potential drawback of the proposed solution, particularly concerning ERC20 token transfers, is the scalability and congestion issues inherent in Ethereum's network, leading to slower transaction times and

increased gas fees during periods of high network activity. This could pose challenges in achieving real-time verification [16], [17]and efficient credit transfer, especially in environments with a large volume of transactions, careful consideration of network limitations and scalability issues, particularly in ERC20 token transfers, is necessary for effective implementation. Based on previous schemes provides by previous papers, this paper mainly focuses on establishing an efficient transfer using ERC-20 standards using safeMath Library to secure smart contracts regarding token transfer.

III. EXISTING SYSTEM

The existing system discussed further identifies certain missing features in the current framework for token management. These include securing token transfers by utilising existing frameworks and libraries on ethereum network and also seamless token integration with wallets and token creation. The extended smart contracts, although providing a solid foundation, lacks the ability to handle these specific time-based events, certified trade pricing, and privacy protection. Furthermore, it acknowledges the limitations in existing security token offerings (STOs) [18] in effectively managing the post-tokenization phase and emphasizes the importance of addressing these issues to unlock the full potential of security tokens. The existing solutions involve enhancing the functionality of smart contracts, particularly through the introduction of extended smart contracts, to handle these missing features and automate the management and operation of security tokens.There was no system built for integrating token transfer across various networks which include testnets and faucets along with local network.The existing system described in [8] focuses on the use of decentralized applications and smart contracts to address the challenges faced by NFT marketplace ventures by the user centric system that closely aligns with the methodology desribed in this paper, but the integration with multiple wallets and networks was not done.To enhance this, we propose the integration of decentralized applications with user interfaces with Solidity, React.js, Web3.js, and Next.js.

IV. METHODOLOGY

Methodology of Decentralized ERC-20 token transfer involves the efficient transfer of tokens using smart contracts that are written based on ERC-20 standards. These smart contracts are integrated with available wallets like Metamask for token transfer.

A simple User Interface (UI) is created using HTML, JavaScript enabling users to use functionalities regarding tokens transfer that include buying and selling of tokens, keeping the limit of token supply and price per each token. These are run in hardhat environment, enabling users to transfer tokens from one network to other.

Users can be able to track their transactions using etherscan.io platform that are done through wallets. The functionalities are enabled through available functionalities in ERC-20 standards such as **balance(), TransferFrom(), balanceOf(), totalSupply().**

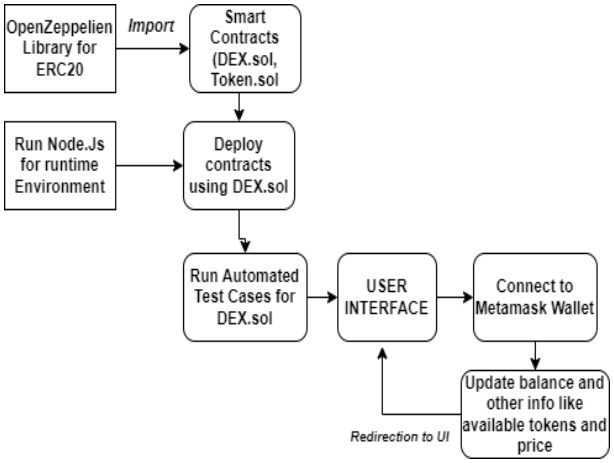

Fig. 1. Workflow of ERC-20 Token Transfer

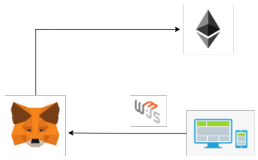

Fig. 2. Token Transfer using Ethereum.

Automated test cases will be authored using the Hardhat environment and JavaScript. These test cases will cover essential functionalities, including selling tokens, buying tokens, getting token prices, granting tokens, and other operations critical for the decentralized exchange. Rigorous testing will ensure the robustness and reliability of the implemented functionalities.

V. IMPLEMENTATION

The implementation involves integrating ERC-20 token functionality and a decentralized exchange (DEX) using OpenZeppelien and the Hardhat development environment. Two smart contracts will be authored – one for the ERC-20 token and another for the DEX with buy and sell functionalities based on set limits.

For the ERC-20 token, OpenZeppelien will be utilized to ensure security and standard compliance. This smart contract will define token properties such as name, symbol, and total supply, along with functions for transferring tokens and checking balances.Also users can have control on price per token.Here, we propose various algorithms that are listed below.

- To sell Tokens
- To withdraw Funds
- To get tokens price
- To buy Tokens
- To check Token Balance

Simultaneously, a smart contract for the DEX will be developed using OpenZeppelien libraries. This contract will facilitate the buying and selling of tokens based on specified limits. The integration of these contracts will form the foundation of the decentralized exchange, providing users with a platform for seamless token trading.

Algorithm 1 Price Calculation Algorithm

function GETPRICE(*numTokens*)
Input: *numTokens* - The number of tokens
Output: *price* - The calculated price
Calculate the price by multiplying the number of tokens with the price.
price ← *numTokens* × *price*
return *price*
end function

Algorithm 2 Token Withdrawal Algorithm

1: **procedure** WITHDRAWTOKENS
2: **Effect:** Withdraw tokens from the contract balance to the owner
3: Retrieve the balance of tokens held by the contract.
4: *balance* ← associatedToken.balanceOf(address(this))
5: Transfer the entire balance of tokens to the owner.
6: ASSOCIATEDTOKEN.TRANSFER(MSG.SENDER, *balance*)
7: **end procedure**

Algorithm 3 Token Buying Algorithm

1: **procedure** BUYTOKENS(*numTokens*)
2: **Input:** *numTokens* (number of tokens to buy)
3: **Output:** None
4: **Validate Token Availability:**
5: Ensure that the number of tokens to buy does not exceed the available token balance.
6: **Require:** *numTokens* ≤ getTokenBalance(); Display error message: "not enough tokens".
7: Calculate the total price for the tokens based on the current price.
8: *priceForTokens* ← getPrice(*numTokens*)
9: **Validate Payment:**
10: Ensure that the amount of Ether sent is equal to the total price for the tokens.
11: **Require:** *msg.value* == *priceForTokens*; Display error message: "invalid value sent".
12: Transfer the specified number of tokens to the buyer.
13: associatedToken.transfer(msg.sender, numTokens)
14: **end procedure**

Algorithm 4 Token Balance Retrieval Algorithm

1: **function** GETTOKENBALANCE
2: **Input:** None
3: **Output:** *balance* - The balance of tokens held by the contract
4: Retrieve the balance of tokens held by the contract.
5: *balance* ← associatedToken.balanceOf(address(this))
6: **return** *balance*
7: **end function**

Algorithm 5 Token Selling Algorithm

procedure SELLTOKENS
Input: None
Output: None
Retrieve the allowance granted by the caller to this contract.
allowance ← associatedToken.allowance(msg.sender, address(this))
Validate Allowance:
if *allowance* ≤ 0 **then**
Display error message: "Insufficient allowance."
Terminate function.
end if
Transfer Tokens:
Initiate the transfer of tokens from the caller to this contract.
sent ← associatedToken.transferFrom(msg.sender, address(this), allowance)
Validate Token Transfer:
if !*sent* **then**
Display error message: "Failed to send tokens."
Terminate function.
end if
end procedure

The Hardhat development environment will be used for testing and deployment. The setup will include installing necessary dependencies, configuring Hardhat for deploying and testing smart contracts, and ensuring compatibility with OpenZeppelien.

To enhance user experience and facilitate transactions, the integration of MetaMask will be implemented. Users will be able to connect their MetaMask wallets to the platform, allowing for secure and convenient interactions with the deployed smart contracts [19], [20]. This integration will be particularly crucial for executing transactions on the Ethereum blockchain.

The user interface (UI) will be developed to connect with the Solidity files and JavaScript for asynchronous communication. The UI will incorporate logic written in Solidity, allowing users to seamlessly engage in token transfers and trades on the DEX. MetaMask will serve as the gateway for users to log in, authorize transactions, and view transaction history.

Finally, the implementation will leverage platforms like Etherscan.io to provide visibility into transactions conducted on the Ethereum blockchain. This will enable users to verify and track their transactions on external platforms, enhancing transparency and accountability. The combination of OpenZeppelin, Hardhat, MetaMask, and a well-designed UI will result in a comprehensive ERC-20 token transfer and decentralized exchange platform [21].

VI. RESULTS

In our test results, we successfully ran various test cases that can be found in figure 5 within the Hardhat environment, highlighting the robustness of our ERC-20 token transfer system. Our emphasis on testing the limitation of token supply showcased the system's reliability and practicality.Figure 3

TABLE II
COMPARISION OF VARIOUS DEPLOYMENT ENVIRONMENTS

Comparision of various Deployment Environments				
Feature	Hardhat (Local Dev)	Polygon Mumbai	Goerli Testnet	Sepolia Testnet
Cost	Free	Low	Medium	Medium
Risk	None	Low	Medium	Medium
Accessibility	Requires setting up Hardhat and tools	Requires access to Polygon Mumbai Faucet and RPC endpoint	Requires access to Goerli node or provider	Requires access to Sepolia node or provider
Testing and Debugging	Easy to test and debug transactions with full control over the environment	Allows testing in a simulated environment close to real-world conditions	Allows testing in a simulated environment close to real-world conditions	Allows testing in a simulated environment close to real-world conditions
Suitability	Ideal for initial development, testing, and debugging	Suitable for testing functionality and integration with other applications on Polygon network	Suitable for testing functionality and integration with other applications on Ethereum network	Suitable for testing functionality and integration with other applications on Ethereum network
Experimental Gas Cost (Gwei)	N/A	5-15	20-50	20-50
Approx. Time Consumption -	Instantaneous -	2-5 seconds -	10-30 seconds -	10-30 seconds -

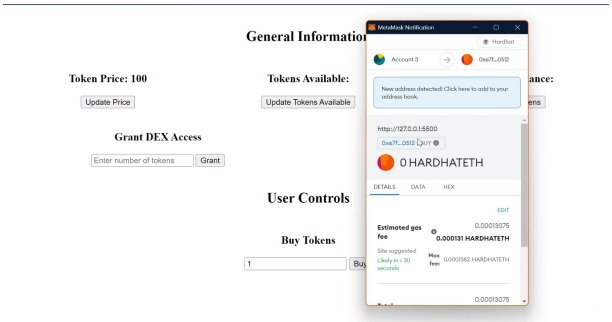

Fig. 3. Buying Tokens using ERC Standards.

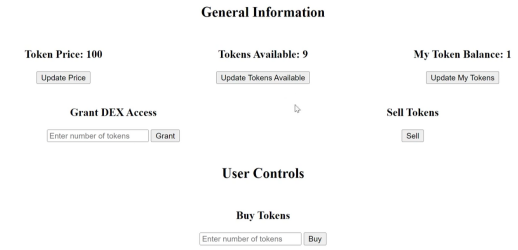

Fig. 4. Updated figures after the transaction in Landing Page.

demonstrates the integration of wallet with the webpage for performing the approval of DEX function in hardhat network.The network can be changed based on user interest by modifying hardhat-config.js file with testnet name and chainId. Here we used chainId 31337 for hardhat. After accessing the tokens for exchange, it is set up for owner to buy the tokens with given token price,and the transaction can be executed using metamask wallet with available test network funds.Figure demonstrates the updated figures after buying tokens.Here we granted 10 tokens for exchange and one token was bought with metamask account,so the remaining tokens will be 9 and updated account tokens will be 1.These figures change based on the metamask accounts and networks.Different accounts on same network will have different token balances which can be compared with two persons belonging to same bank but they dont have same money in account,but final balance of bank will be same for both.Likewise, remaining tokens for DEX will be same for all accounts of same wallet in same network.Likewise we can also add withdraw and adding funds option for only owner or set of accounts which helps build Intial Coin Offerings and Staking platforms. Connecting to MetaMask, we demonstrated seamless token transfers between different MetaMask accounts, confirming the user-friendliness

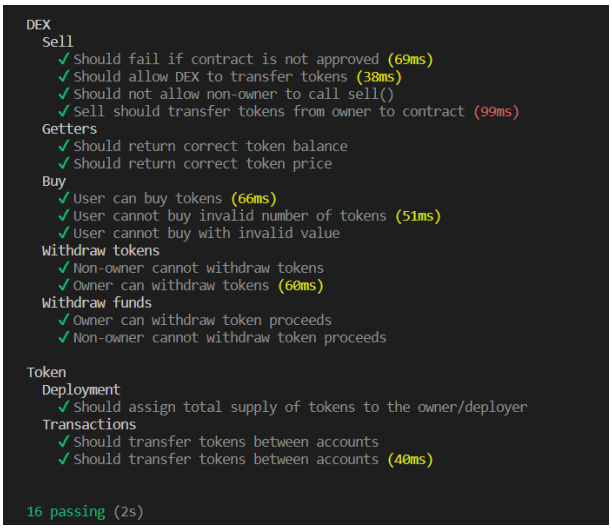

Fig. 5. Passed Test Cases for ERC-20 tokens transfer.

of our solution. Our focus on controlling token supply underscores our commitment to creating a secure ecosystem. Table III provides the previleges for owner as well as users with certain functionalities.

TABLE III
TEST CASES- OWNER AND USER

Owner test cases				
Approve	Sell Tokens	Buy Tokens	Withdraw Tokens	Withdraw Funds
Yes	-	-	Yes	Yes
User test cases				
Yes	Yes	Yes	-	-

These positive results show that our decentralized web application works well in real-world situations.These findings give us a solid base to improve and optimize our system. We are on the way to make our solution an efficient way for people to transfer tokens on the blockchain.

Here we have provided the comprehensive analysis of the test cases for various functionalities and privileges that both user and owner has while interacting with the Token Transfers in Hardhat development Environment.

VII. CONCLUSIONS AND FUTURE SCOPE

In summary, the development of a token transfer system based on the ERC-20 standard within ethereum platform revolutionized financial technology by transferring digital assets. This encapsulates decentralization, transparency, and real-time efficiency, reshaping traditional approaches to financial transactions.The inherent features of blockchain, including immutability and decentralized consensus, play a pivotal role in fostering trust among participants in the self-sustaining token-based ecosystem, fostering continuous user engagement and sustainable growth.As we look towards the future, there's an acknowledgment of the need for continued evolution. To address this, we aim to implement limitations on the total supply of ERC-20 tokens from the user side, enhancing control and stability within the ecosystem.

Furthermore, recognizing the importance of optimizing gas fees, efforts will be directed towards increasing efficiency in transactions, ensuring cost-effectiveness, and providing a more seamless experience for users. These strategic enhancements elevates the overall functionality of the ERC-20 token transfer system, paving the way for a more inclusive and resource-efficient financial landscape.However the scope is multidimensional. Opportunities include expanding the range of supported tokens, integrating with decentralized finance (DeFi) protocols, exploring cross-chain compatibility, and continual enhancement of smart contracts for security and scalability.

Moreover, prioritizing user-centric innovations, such as more sophisticated financial instruments and improved user interfaces and addressing environmental concerns through sustainable blockchain practices further contribute to the project's long-term viability and its adaptability to the ever-evolving landscape of decentralized finance.Finally this paper presents a view of ERC-20 based transactions that are user centric in order to promote blockchain in DeFi space, however we have not considered onchain concerns to our system which may include dynamic gas fee, network congestion when user scale is high and consensus mechanism which provide strong connectivity of the network without any crashes and there is a scope to improve efficiency in terms of saving gas fee, prioritizing transactions based on gas fee and speeding up transactions in ethereum Network.

REFERENCES

[1] V. Buterin, et al., A next-generation smart contract and decentralized application platform, white paper 3 (37) (2014) 2–1.

[2] M. Shirole, M. Darisi, S. Bhirud, Cryptocurrency token: An overview, in: IC-BCT 2019: Proceedings of the International Conference on Blockchain Technology, Springer, 2020, pp. 133–140.

[3] M. Soni, E. Gandotra, Erc-20 token exchange system over blockchain network (2023).

[4] P. Shamili, B. Muruganantham, Blockchain based application: Decentralized financial technologies for exchanging crypto currency, in: 2022 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI), IEEE, 2022, pp. 1–9.

[5] Y. Wang, K. Li, Y. Tang, J. Chen, Q. Zhang, X. Luo, T. Chen, Towards saving blockchain fees via secure and cost-effective batching of smart-contract invocations, IEEE Transactions on Software Engineering 49 (4) (2023) 2980–2995.

[6] S. Fujimoto, K. Omote, Proposal of a smart contract-based security token management system, in: 2022 IEEE International Conference on Blockchain (Blockchain), IEEE, 2022, pp. 419–426.

[7] G. Gan, E. Chen, Z. Zhou, Y. Zhu, Token-based access control, IEEE Access 8 (2020) 54189–54199.

[8] S. R. Behera, D. P. Mishra, S. Parida, A. R. Patro, Proposal of user-friendly design of nft marketplace, in: 2023 4th International Conference on Computing and Communication Systems (I3CS), IEEE, 2023, pp. 1–6.

[9] J. Priest, C. Cooper, S. Lovell, Y. Shi, D. Lo, Design and implementation of an erc-20 smart contract on the ethereum blockchain, in: 2023 IEEE International Conference on Big Data (BigData), IEEE, 2023, pp. 2334–2338.

[10] S. A. Jaffer, S. Pandey, R. Mehta, P. Bhavathankar, Blockchain based direct benefit transfer system for subsidy delivery, in: 2020 international conference for emerging technology (INCET), IEEE, 2020, pp. 1–6.

[11] J. R. Gudeme, S. K. Pasupuleti, R. Kandukuri, Attribute-based public integrity auditing for shared data with efficient user revocation in cloud storage, Journal of Ambient Intelligence and Humanized Computing 12 (2021) 2019–2032.

[12] J. R. Gudeme, S. K. Pasupuleti, R. Kandukuri, Certificateless multi-replica public integrity auditing scheme for dynamic shared data in cloud storage, Computers & Security 103 (2021) 102176.

[13] J. R. Gudeme, S. Pasupuleti, R. Kandukuri, Certificateless privacy preserving public auditing for dynamic shared data with group user revocation in cloud storage, Journal of Parallel and Distributed Computing 156 (2021) 163–175.

[14] D. Li, W. E. Wong, J. Guo, A survey on blockchain for enterprise using hyperledger fabric and composer, in: 2019 6th International Conference on Dependable Systems and Their Applications (DSA), IEEE, 2020, pp. 71–80.

[15] S. Narongsak, S. Nonsiri, Digital token: Token-based degree certificates with credit transfer system, in: 2023 8th International Conference on Business and Industrial Research (ICBIR), IEEE, 2023, pp. 1312–1317.

[16] Z. He, Z. Liao, F. Luo, D. Liu, T. Chen, Z. Li, Tokencat: detect flaw of authentication on erc20 tokens, in: ICC 2022-IEEE International Conference on Communications, IEEE, 2022, pp. 4999–5004.

[17] S. Goswami, R. Singh, N. Saikia, K. K. Bora, U. Sharma, Tokencheck: towards deep learning based security vulnerability detection in erc-20 tokens, in: 2021 IEEE Region 10 Symposium (TENSYMP), IEEE, 2021, pp. 1–8.

[18] T. Lambert, D. Liebau, P. Roosenboom, Security token offerings, Small Business Economics (2021) 1–27.

[19] M. Alharby, A. Van Moorsel, Blockchain-based smart contracts: A systematic mapping study, arXiv preprint arXiv:1710.06372 (2017).

[20] M. Alharby, A. Aldweesh, A. Van Moorsel, Blockchain-based smart contracts: A systematic mapping study of academic research (2018), in: 2018 International Conference on Cloud Computing, Big Data and Blockchain (ICCBB), IEEE, 2018, pp. 1–6.

[21] F. Schär, Decentralized finance: On blockchain-and smart contract-based financial markets, FRB of St. Louis Review (2021).