Blockchain Gaming-A Basic Exploration and Development Insights

Luohaotian He

International School, Beijing University of Post and Telecommunication, Beijing, 100876, China
heluohaotian@bupt.edu.cn

Abstract. Currently, blockchain technology is finding applications across a broad range of industries beyond its traditional role as a decentralized ledger. A notable area of expansion is in blockchain gaming, which leverages the core attributes of blockchain—transparency, immutability, and decentralization—to address the trust issues traditionally associated with game operators. This paper explores various existing examples of how blockchain technology has been integrated into digital gaming. For a more practical exploration, the author developed a blockchain-based version of the game Monopoly. This adaptation specifically addresses the generation of random numbers, a complex and critical challenge within the blockchain framework. Additionally, the paper delves into other essential facets of blockchain game development, including the design of NFTs(Non-Fungible Tokens), the creation of security mechanisms for smart contracts, and strategies for optimizing gas usage. These developments underscore the potential of blockchain to transform gaming dynamics by enhancing fairness, security, and player ownership.

Keywords: Blockchain, Smart contract, Security

1 Introduction

Since Satoshi Nakamoto introduced the seminal Bitcoin white paper in 2008, blockchain technology has garnered immense interest from researchers, governments, and enterprises [1]. The advent of Ethereum brought about smart contracts, heralding the era of Blockchain 2.0 and enabling the development of decentralized applications (DApps) [2]. Notably, blockchain games now constitute over 50% of transactions on platforms such as Ethereum and EOS(EOSIO), underscoring their significance within the DApp ecosystem. However, the blockchain landscape faces challenges, including a scarcity of transformative applications and the prevalence of ICO(Initial Coin Offering) scams and devalued "air tokens," which introduce uncertainty into the industry.

© The Author(s) 2024
Y. Wang (ed.), *Proceedings of the 2024 2nd International Conference on Image, Algorithms and Artificial Intelligence (ICIAAI 2024)*, Advances in Computer Science Research 115,
https://doi.org/10.2991/978-94-6463-540-9_30

The main body of this paper is structured into three key sections: First, it commences with a comprehensive review of the existing literature and research on various aspects of blockchain gaming. This section provides insights that facilitate a basic understanding of the mechanisms driving blockchain games. Next, the paper details a practical endeavor where the author developed a Monopoly game using blockchain technology. This case study exemplifies the potential applications of blockchain in the gaming sector, illustrating practical implementations and exploring how blockchain can revolutionize game dynamics through enhanced transparency and player autonomy.

2 Related Work

2.1 Blockchain Game

In this paper, blockchain games can be generally defined as a series of digital games designed and implemented based on the nature of blockchain technologies. And blockchain games can be divided into 4 categories, based on advantages blockchain technology offer. As shown in Fig.1.

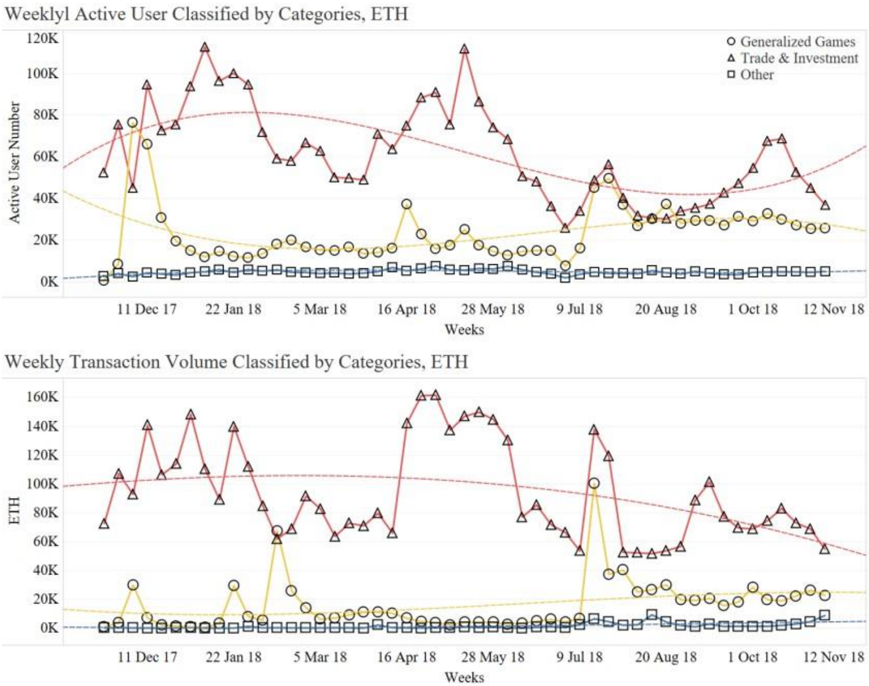

Fig. 1. DApps on Ethereum**.**

Rule Transparency. Nothing like traditional games, blockchain-based games provide access to its basic rules and operations for any third-party organizations or players, firmly building up their reliability.

Example 1: Satoshi Dice; Bitcoin; 4/2012.

Example 2: FOMO 3D; Ethereum; 7/2018.

Asset Ownership. In the case of traditional online games, all digital properties belong to the single entity-game operators since all data is kept on operators' servers. However, it becomes feasible for game players to truly own their game assets with the help of NFT technology to link these assets to players' own addresses.

Example 1: CryptoKitties; Ethereum; 11/2017.

Example 2: Gods Unchained; Ethereum; 7/2018.

Asset Reusability. The essence of blockchain lies in its distributed ledger technology, where data is organized into a chain of blocks linked together.

In essence, the blockchain is an open-source ledger in which data and smart contracts are organized into a chain of blocks linked together. Considering this, blockchain games may leverage a customized framework to incorporate business data from existing games. And a new gaming ecosystemacross different games and blockchain platforms could be built, incentivizing players to participate more actively in games' trade system.

Example 1: KittyRace; Ethereum; 3/2018.

Example 2: KotoWars33; Ethereum; 12/2018.

User-Generated Content (UGC). Traditionally, ownership of user-generated content (UGC) belongs to the game operators. But as was mentioned in the part Asset Ownership above, this advantage will further encourage voluntary developers to produce innovative contents.

Example 1: Cell Evolution; Nebulas; 5/2018.

Example 2: CardMaker37; Nebulas; 7/2018.

Moreover, the blockchain game can also be classified into 2 categories based on the degree of application of blockchain: one is fully decentralized" onchain game", while the other choose to assign certain operations of traditional games to smart contracts on the blockchain or directly store business data on the blockchain [4]. As shown in Fig.2.

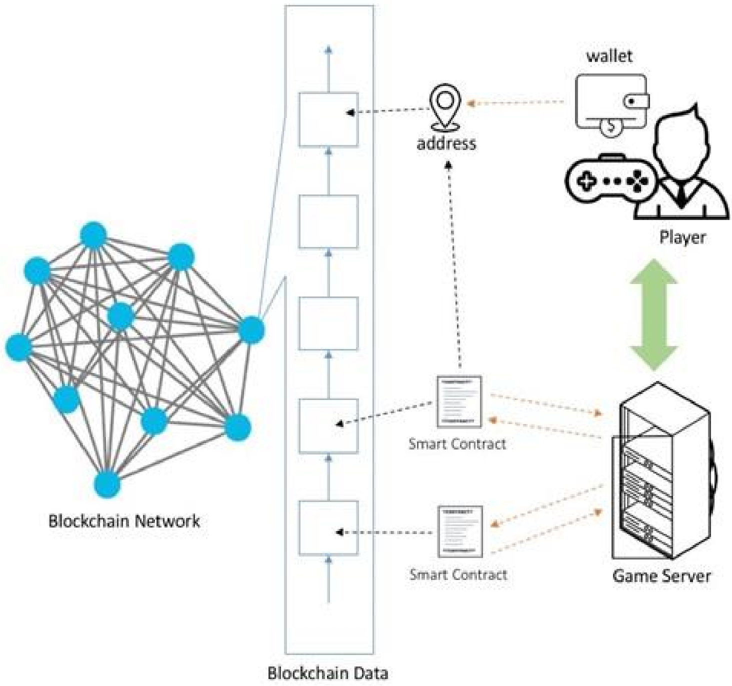

Fig. 2. Architecture for Blockchain Games.

2.2 Smart Contract Vulnerability

Currently, the implementation of smart contracts heavily depends on the decentralized Ethernet virtual machines and the programming language like Solidity [6]. Although most smart contracts will be compiled into bytecodes, it is possible to reverse the process with the help of some tools. And vulnerabilities could be present in many aspects including the Ethereum Virtual Machine (EVM) design, the Solidity language, and the execution logic, etc.

2.3 Auditing Tools

Due to blockchain's immutability, smart contract auditing is crucial due to the challenges posed by patching updates post-deployment. Due to difficulties of patching updates brought by blockchain's immutability, smart contract auditing is crucial. As of today, there exist some proven tools such as Mythril, MythX, SmartCheck, and Delligence Fuzzing.

3 Game Design and Operation Procedure

3.1 Deployment and Operating Environment

To simplify the development process, the author chose to use the general user interface of RemixIDE directly instead of introducing Web3.js to build an interactive interface. Additionally, the author initially attempted to run the contract on a private chain using Metamask with Ganache, but debugging and running were too cumbersome. As a result, the entire development process was eventually conducted on Remix's virtual chain.

3.2 Modual Design

To create a Monopoly game with blockchain technology named Blockpoly, four modules are needed. The following Fig.3 is the graph of four modules in Blockpoly (a combined name with blockchain and Monopoly).

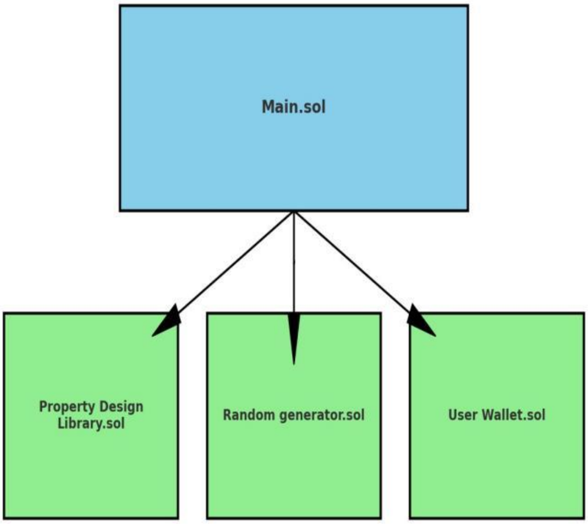

Fig. 3. Module Design.

3.3 Functions

The Table 1-4 demonstrate the detailed function designs within each module of Blockpoly.

Table 1. Functions in event. Sol.

Function	Description
User_ Init ()	Initializes the user's wallet with an initial capital of 500.
Throw_ Dice ()	Simulates rolling a dice and determines the outcome based on the player's current position.
Draw_ Event ()	Draws a chance event assisted with R Event ().
property_ buy ()	Allows the player to buy a property if the player arroved at that specific grid.
toString ()	Converts a uint to a string.
Address ToString ()	Converts an address to a string.
Char ()	Converts a bytes1 to a string.

Table 2. Functions in propertynft. Sol.

Function	Description
create_ Property ()	Creates a new property NFT and assigns its ownership to a specified user.
Transfer Property ()	Transfers the ownership of a property NFT.
Upgrade Property ()	Upgrades a property, increasing the toll fee.
Get User Property ()	Retrieves the properties owned by a specified user.
Is Property Owner Exists ()	Checks if a specified property has an owner.
Get Data ()	Retrieves data of a specified property, including name, purchase price, toll fee, and owner.

Table 3. Functions in random_ generator. Sol.

Function	Description
R_ Dice ()	Returns a random number between 1 and 6 to simulate the result of rolling a dice.
R_ Event ()	Returns a random number between 0 and 9 to select the type of event.

Table4. Functions in wallet. Sol.

Function	Description
Deposit ()	Deposits a specified amount into the user's wallet.
Withdraw ()	Withdraws a specified amount from the user's wallet.
Get Balance ()	Retrieves the balance of a specified user.
Get Position ()	Retrieves the player's position on the game board.
Update Position ()	Updates the user's position on the game board, resetting it to the new position and rewarding a certain amount if the user exceeds a certain limit.
Initialize ()	Initializes the user's wallet with a default balance.

3.4 Board Design

As shown in Fig.4, the board is similar to the real Monopoly game with 32 squares in total—1 starting point, 24 property grids and 7 event grids.

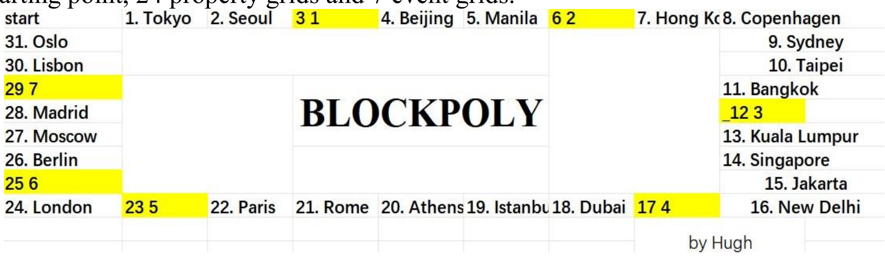

Fig. 4. Board of blockpoly .

3.5 Operation Procedure

The game progresses through following steps:

Initialization: The game initializes with the initial_ user () function, setting up player's initial account. Each player receives an initial fund of 500.

Throw Dice: Each player uses the throw dicebuttontoroll the diceinturn.

The result determines player's movement on the chessboard. Landing on a property grid allows the player to consider purchasing the property. If landing on an event grid, the player needs to draw chance cards via the draw_ event function.

Property Purchase: Players could purchase properties using the property_ buy function if allowed. Funds are transferred from the player's balance to the property owner, granting ownership.

Event Drawing: Events may result in deposit or withdrawal of funds from players' balance.

Toll Payment:

A toll cost is paid after landing on another player's property. The toll fee is added to the property owner's balance and subtracted from the player' s.

Ending Condition:

The results of dice rolls determine how the game progresses step by step. Players receive a 100 as reward upon passing the starting point, added to their balance. Once the player's assets have reached 5,000, the game stops and gives the message, you win!"

4 Evaluation and Critical Assessment

4.1 NFT Design

When designing the property module, the author utilized the ERC721 library from OpenZeppelin. After integrating the ERC721 library into the project, it becomes capable of interacting with other applications and services that are compatible with ERC721. Not only that, the security of NFTs is also ensured.

4.2 Generation of Random Number

There are basically two ways in generation of random numbers in blockchain [7].

On-chain Generation. a) On-chain Generation: On-chain generation relies on selecting unpredictable seeds before transactions are packed into blocks. Utilizing block attributes like block. timestamp and block. difficulty as seeds enhances unpredictability. Repeating hash operations on the initial random numbers increase security by making attacks more costly. Additionally, incorporating business data into the random number generator, such as player addresses and selected numbers (in lottery contracts) is also a reasonable way.

Off-chain Generation. The generation of random numbers can use off-chain oracles like Chainlink to connect to third–party organizations like Randao and Niguez Random Engine for on-chain use. And in project Blockpoly, the author employed a pseudo-random approach, using block. timestamps with block. prevrandao as seeds.

4.3 Safety Design

Here are several vulnerabilities that developers may run into during the development of blockchain games, along with corresponding solutions [8].

Reentrence. Oweing to solidity's unique fallback mechanism, reentrancy is a serious issue when smart contracts deal with sensitive operations such as transfers. To mitigate this risk, one must carefully verify the contract author and properly schedule the order of external calls and internal code execution [9].

Overflow/Underflow. In games, character attributes such as attack, defense, and health may change regularly owing to this. Developers can address integer overflow risk by either validating arithmetic operations or using the OpenZeppelin's SafeMath library for secure arithmetic [10].

Denial of Service (DoS). The default approach is to activate the fallback and recovery mechanism, rather than being triggered by the gas limit. Design relevant functions carefully to avoid giving attackers the opportunity to create infinite loops.

Predictable Variable. Sometimes, variables that generated by internal functions are predictable. And when they assist core functions of DApps, these DApps are susceptible to malicious attacks. One possible solution is to find some confidential ways instead as discussed in Generation of Ramdom Number.

4.4 Gas Consumption Optimization

Gas consumption and optimization are not seriously taken into account during the programming process, but it is an indispensable part in practical development.

Gas Consumption Analysis. By utilizing the gas estimation feature of the Solidity compiler, developers can accurately assess gas consumption and identify potential high-consumption areas, providing crucial reference for subsequent optimization work.

Gas Optimization Strategies. In this section, three effective gas optimization strategies are proposed. Firstly, optimizing algorithms and logic to reduce unnecessary computations and operations is essential. Secondly, adopt some strategies to avoid loops and iterations to minimize the gas consumption. Lastly, reduce the demand for storage and read operations, aiming at lower gas consumption and improving the execution efficiency.

Performance Testing. Performance testing in gas consumption optimization is needed. Potential bottlenecks and problems can be found by thoroughly examining contract performance under varied gas consumption situations.

5 Conclusion

The project Blockpoly represents a practical exploration into the integration of blockchain technology within gaming. Optimization of the system can be achieved in three key areas: (1) Enhancing the generation of reliable random numbers for on-chain use by incorporating business data, employing multiple hashing, or utilizing an off-chain oracle to ensure randomness. (2) Implementing the OneZeppelin's SafeMath library to prevent numerical errors. (3) Using web3.js to develop a user-friendly interactive interface that encapsulates the program effectively.

This paper examines various dimensions of blockchain game development, including the implementation of smart contracts, the generation of random numbers, security analysis, and optimization strategies. It provides insights into the practical applications and inherent challenges within the blockchain gaming industry, contributing valuable knowledge to developers and stakeholders engaged in this evolving field.

References

1. Nakamoto, S., "Bitcoin: A peer-to-peer electronic cash system," https://bitcoin.org/bitcoin.pdf (2008).
2. Mohanta, B. K., Panda, S. S., & Jena, D., "An overview of smart contract and use cases in blockchain technology," Proceedings of the 2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT), pp. 1-4 (2018).
3. Min, T., Wang, H., Guo, Y., & Cai, W., "Blockchain games: A survey," Proceedings of the 2019 IEEE Conference on Games (CoG), pp. 1-8 (2019).
4. Cai, W., & Wu, X., "Demo abstract: An interoperable avatar framework across multiple games and blockchains," Proceedings of IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp. 967-968 (2019).
5. Du, M., Chen, Q., Liu, L., & Ma, X., "A blockchain-based random number generation algorithm and the application in blockchain games," Proceedings of the 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC), pp. 3498-3503 (2019).
6. Woehrer, M., & Zdun, U., "Smart contracts: Security patterns in the ethereum ecosystem and solidity," Proceedings of the 2018 IEEE 1st International Workshop on Blockchain Oriented Software Engineering (IWBOSE), pp. 2–8 (2018).
7. Liu, T., "Several methods for generating random numbers on ethereum," https://learnblockchain.cn/article/1083 (2020).

8. Qian, P., Liu, Z. G., He, Q. M., Huang, B. T., Tian, D. Z., & Wang, X., "Smart contract vulnerability detection technique: A survey," Journal of Software, vol. 33, no. 8, pp. 3059-3085 (2022).
9. Min, T., & Cai, W., "A security case study for blockchain games," Proceedings of the 2019 IEEE Games, Entertainment, Media Conference (GEM), pp. 1-8 (2019).
10. Yuen, H. Y., Wu, F., Cai, W., Chan, H. C., Yan, Q., & Leung, V. C., "Proof-of-play: A novel consensus model for blockchain-based peer-to-peer gaming system," Proceedings of the 2019 ACM International Symposium on Blockchain and Secure Critical Infrastructure, pp. 19-28 (2019).

Open Access This chapter is licensed under the terms of the Creative Commons Attribution-NonCommercial 4.0 International License (http://creativecommons.org/licenses/by-nc/4.0/), which permits any noncommercial use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.