THE UNIVERSITY OF HONG KONG

DEPARTMENT OF COMPUTER SCIENCE

Gcoin Trading Platform

A Gaming Assets Trading Platform Based on Blockchain Technology

Final Report

Supervisor: Dr. CUI Heming

YANG Yuening
3035448926

Date of submission: 2021-04-18

Abstract

Gaming trading is popular today and continues to keep a high speed of growth. However, problems exist in the current gaming market. First, the liquidity of gaming currencies is inferior across games. Second, there is no trusting platform that publishes the transaction data and where users can learn the revenue data for merchandise sales in the game. Third, game companies and trading platforms usually have a long revenue settlement period. Forth, the gaming platform hijacks game developers with user traffic and charges them high fees.

According to these defects, our project implemented a gaming assets trading platform, Gcoin Platform. It is implemented based on the Ethereum blockchain, and companies can join the forum by adding their nodes to the blockchain network. Because of the blockchain features, users can easily get the voucher of their transactions, and the companies are more trustable as their profitability is accessible to all. Oracle cross-chain algorithm is applied to ensure that companies can withdraw earnings at any time. As all manipulate the underlying blockchain, there is less need for high transaction fees. Moreover, since the Gcoin Platform offers a reliable assets exchange platform, it is possible to create a digital currency that circulates across games, creating a shared and healthy game economy.

There are several methods in the platform's development process. Two chains are involved. The first one is Ethereum mainnet, a famous public blockchain with significant user traffic and crypto-currency reserves. The second is a permissioned chain implemented on Etheruem, with EGES applied as its consensus algorithm. The nodes of platform supporters and all companies constitute the private chain on the Gcoin Platform. All business work is on the permissioned chain for its higher throughput. A cross-chain protocol is used to handle cross-chain deposits and withdrawals. The system has a front-end in Vue framework to directly interact with two blockchain networks to ensure decentralization and a back-end written in Java to manage some business logic.

Acknowledgment

I would like to express my deep thankfulness to my advisor Prof. Cui, who gave me a chance to study an exciting topic and offered us a consensus algorithm, EGES, which improved our Gcoin Platform’s safety and efficiency. I am also willing to sincerely thank Mr. Chen, a Ph.D. of Prof. Cui, who gave us helpful suggestions and strongly supported us on technical difficulties encountered. I would also like to thank my teammates, Nicholson and Tracy. They have paid a lot in this project and we have a very good cooperation.

Table of Contents

1 Introduction

Our project is to build an assets trading platform with API for in-game transactions. The platform is based on the blockchain, and the immutability and decentralization features of the chains empower the Gcoin Platform to become a trustworthy and secure party. Instead of individual gaming points used in current games, our platform uses “Gcoin” for in-game purchases. Since Gcoin is a cryptocurrency released on the chain, all transactions related to it are also recorded in the chain. Users can easily get vouchers for their in-game trading, and the games are more trustworthy as their profit from those transactions is accessible to the public. A permissioned chain is implemented to increase the overall transaction throughput, and Oracle cross-chain mechanism is applied to ensure the safety and reliability of the permissioned chain. The reliable Gcoin Platform also offers a digital currency that can circulate across games, which creates a shared and healthy game economy. We wish to give a hand to the ecosystem of the virtual world of the games.

1.1 Background

The gaming industry is growing at a rapid pace. According to SuperData, a famous industry analysis firm that deals with gaming-related data, though many industries shrank during the pandemic, the game industry has grown 12% in 2020, from $120.1 billion in 2019 to $139.9 billion [1]. In the global market game report published on NewZoo, the author predicted that the number of players in the game mark would still maintain an average growth rate of 7.7% from 2018 to 2023, reaching $200.8 billion (See Figure 1) [2].

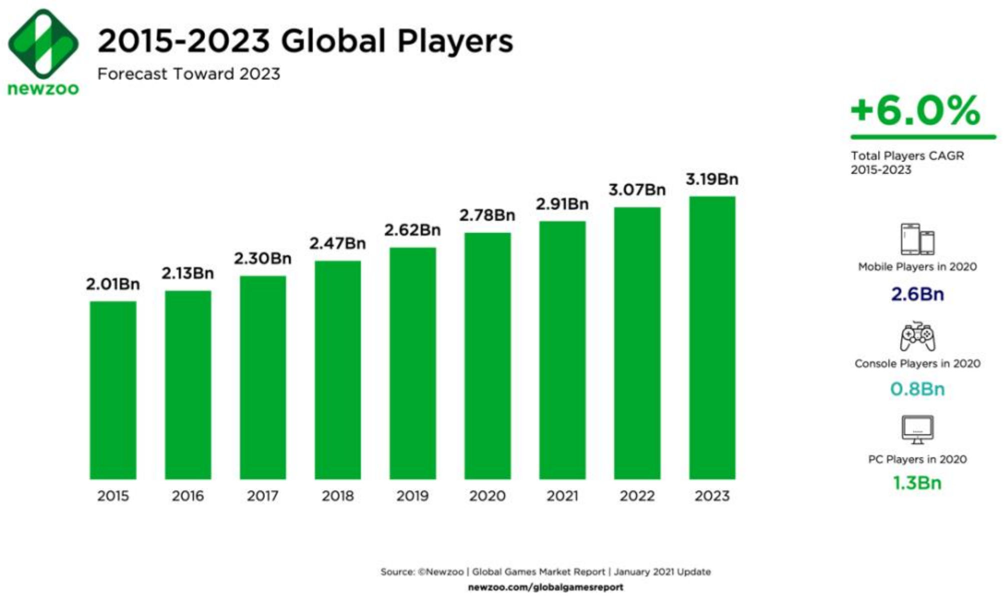


Figure 1. The number of global players continues to grow from 2015 to 2023 [2]

“Free to play and pay to win” has become a popular strategy for gaming companies to attract users and make a profit. Players are induced to consume gaming products and equipment to power up their gaming characters and enjoy better service. In 2020, free-to-play games accounted for the vast majority of the digital games’ revenue (78%), stated by SuperData [1].

Though the gaming market has a bright future, several problems remain to be solved. First, most games introduce their currency for gamers to perform in-game trading. Such coins are of inferior liquidity, and if players want to switch to another game, all the tokens are left in the old game. This inferior liquidity is a barrier for users to change games and is terrible for the overall game ecosystem. Second, gaming companies’ transaction and profit details are usually hidden from the public, making them less trustworthy. Even if some companies publish their financial report, there is a lack of a reliable platform for users to trust, as data can be modified to cheat people. Third, small game developers usually rely on large gaming platforms, such as Steam, Apple Store, Google Store, etc., to promote their products. However, the revenue settlement period on these platforms is usually long, and companies have to wait for a long time to get their earnings. Moreover, since small games rely on the famous gaming platform to attract new users, they hijack them with high fees. For example, Steam charges USD 100 for each game putting on their platform [3]. Apple charges an annual fee of USD 99 for each developer for holding their app on the shelves. An extra 30% of the revenue will be taken if there is any in-game purchase [4]. All these problems lie like landmines, waiting to be solved.

1.2 Project Idea

We hypothesize a significant need for a platform that can solve those landmines listed before, which benefits not only a single game developer or a single gamer but the whole game ecosystem. More specifically, this paper proposes that a platform based on blockchain technology can essentially meet this requirement.

All transactions on the Gcoin Platform are sent and recorded in the underlying blockchain maintained by multiple merchants who release games on the platform. Therefore, it ensures the platform's accessibility and reliability. To obtain a more significant throughput that may better fit the need of a multi-trading gaming assets exchange platform, safe cross-chain technology and a permissioned chain with a faster consensus algorithm is applied, increasing the concurrent processing capacity of the platform more than five times higher than the average blockchain on the market.

The platform mainly offers two services. Game players can choose to sell the old assets or buy new assets from other players, while gaming companies will take 35 percent of the assets' original price as fees (see Figure 2). Stable coins are used to exchange with Gcoin, a crypto-currency released by the platform and used in all in-game purchases. An on-chain receipt is generated by the underlying blockchain and is sent automatically to the companies' back-end in each payment. The companies verify the receipt, wait for the transaction to be confirmed, and transmit the gamer's specified gaming assets. The platform will offer several interfaces to judge whether a specific product can be sold among players.

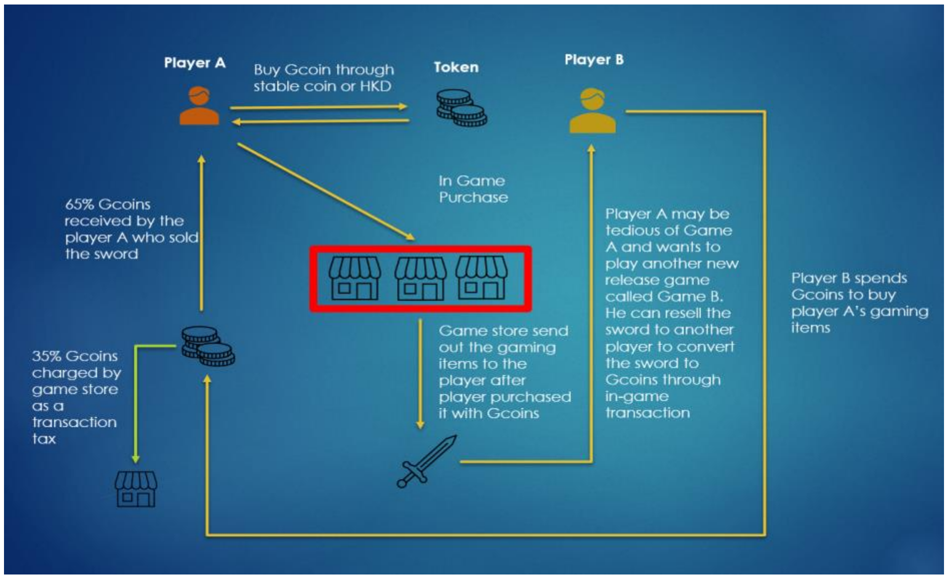

Player A
Buy Gcoin through stable coin or HKD
Token
Player B
In Game Purchase
65% Gcoins received by the player A who sold the sword
Player A may be tedious of Game A and wants to play another new release game called Game B. He can resell the sword to another player to convert the sword to Gcoins through in-game transaction
Player B spends Gcoins to buy player A's gaming items
Game store send out the gaming items to the player after player purchased it with Gcoins
35% Gcoins charged by game store as a transaction tax

Figure 2. An overview of trading on the Gcoin Platform

Both players and companies are able to obtain the platform token, Gcoin using e-payment or the stable coins on Ethereum mainnet. However, the platform restricts that only gaming companies and platform admins are allowed to perform a withdrawal, which reduces customer mobility and attracts more players to the platform. With the decentralized blockchain's help, such leave is not controlled by a single entity, e.g., the platform developers, and thus the game owners do not bear extra cost for trusting the Gcoin Platform. As long as the withdrawal requirements are met, the company can withdraw its money (see Figure 3).

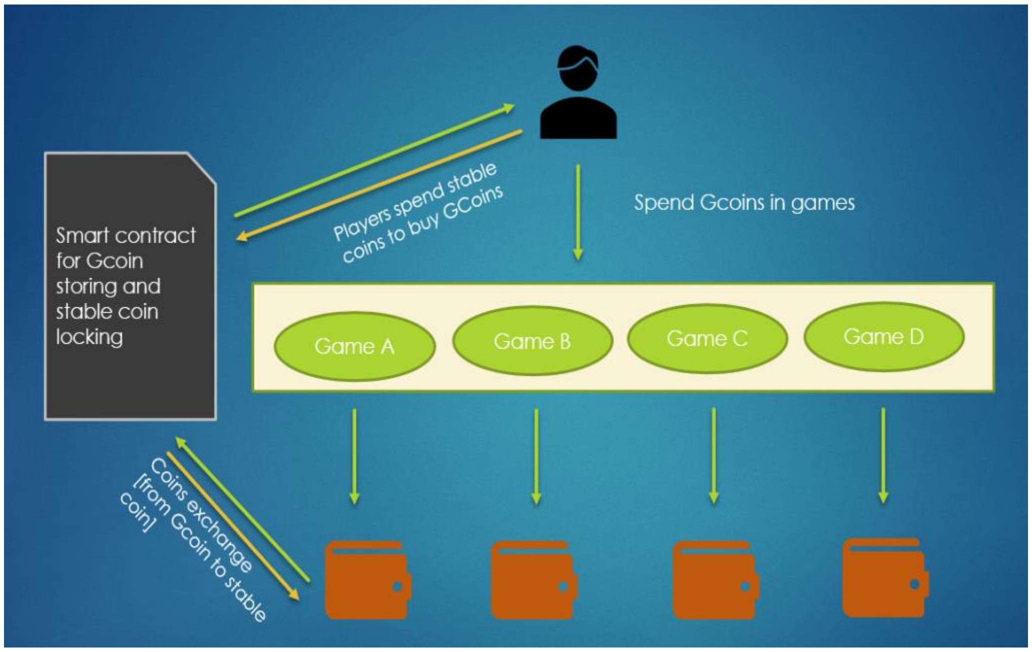

Smart contract for Gcoin storing and stable coin locking
Players spend stable coins to buy GCoins
Spend Gcoins in games
Game A
Game B
Game C
Game D
Coins exchange [from Gcoin to stable coin]

Figure 3. An overview of Gcoin recycling

1.3 Deliverables

The platform font-end is delivered as a website wrapped in a desktop application. We recommend users to access our service through the desktop application, but webpages are also provided for better mobility.

Several services are provided, including wallet service, purchasing using Gcoin, Gcoin top-up, transaction notification, and a child-chain explorer accessible to all gamers and game companies. Special services, such as Gcoin withdrawal, game issuing, and a real-time child-chain monitor.

1.3.1 Wallet Service

The platform offers an embedded wallet for users to perform transactions on the child chain. All users should set their private key before any transaction starts, and their public address will be automatically set. After each transaction or page reloading, the balance and the gas information of the account will be updated and shown.

1.3.2 Purchasing

Gamers can use Gcoin to either buy the newly issued games on the platform or perform in-game transactions. Whenever a new transaction is created, sent to the child chain, and get confirmed on the chain, the application will send the transaction hash to the game providers, who verify the transaction obtained from the chain using the transaction hash, ensure that it is confirmed, and not sent previously, and then transfer the corresponding assets to gamer's gaming account.

To enable in-game purchase, the game developers should implement the API provided by our platform to realize the actions introduced before. Game companies can also add their constraints to limit the transactions against particular assets.

1.3.3 Deposit and Withdrawal

All users can buy Gcoin using USDT on Ethereum mainnet. The process can be complex and involved multiple operations, but all those actions are handled by the application itself and are enclosed in a single button. For gamers who are not familiar with Ethereum, the Gcoin Platform also provides an e-payment service to buy Gcoin through real-world money.

Only registered game companies and platform owners are able to withdraw money from the platform. This restriction is applied to the underlying child chain. Therefore, even if unauthorized entities bypass our application testing and directly interact with the chain, there is no way to perform the withdrawal. All Gcoin will be exchanged to USDT, a stable coin on Ethereum mainnet, and companies can easily change those tokens back to U.S. dollars.

1.3.4 Game Issuing

Companies are able to publish their games on the platform. Information, such as the name, the description of the game, the issuing companies, and the game trailer, should be included.

1.3.5 Notification, Explorer and Monitor

The desktop application provides a notification system and will notify users whenever the transaction is created, sent, and confirmed.

An explorer on the child chain is implemented so that users can see the latest block and transactions and query for the specific transaction as well. The transaction records of the Gcoin payment system are transparent and open to the public, and users can trace back all transaction records in the application.

A real-time monitor is provided for the gaming companies so that they can know the real-time blockchain situation and make adjustments in time.

1.4 Outline for the Remaining Part

The remainder of this paper will proceed as follows. First, the essential methodologies applied in the implementation of the platform will be introduced in section 2, including the dual-chain structure, cross-chain strategy, node adding policy, and overall system design. The detailed implementation will come next in section 3. In this section, different trails of cross-chain strategies, together with their limitations, will be explained. It will also talk about how front-end and back-end interact with the chains. It will mainly focus on the blockchain implementation, the smart contracts design, and the interaction with the chains, which are the main parts the author developed in the project. Section 4 presents the future plan of the platform, followed by a summary in section 5.

2 Methodology

This section now introduces the leading techniques and system structure of the Gcoin Platform. The selection of the home chain and child chain will come first, followed by the design of the permissioned child chain. After that, Oracle cross-chain method will be introduced. Next, the child chain joining policy will be discussed, and finally, the report will show the 4-level system design and talk about the front-end and back-end.

2.1 Blockchain Selection

A Blockchain is a distributed ledger hold by every node in the chain, recording all transactions happening on it. A blockchain is decentralized, immutable, and transparent.

Since all entities involved in the chain keep and manage a copy of the ledger, they can decide whether they accept the new transaction, i.e., all decisions must be made by more than a single entity. The emission of the central institution gives several advantages. First, it reduces the cost of trust. As long as the majority is good, the bad transactions can be ruled out. Second, since the ledger is copied and kept by different entities, it is easier to keep the transaction data transparent to the public. Third, there is no high transaction fee decided by the central authority. Rather than that, all nodes capable of producing new blocks to store the transactions decide the transaction fees together.

The immutability of the blockchain is guaranteed by the techniques of cryptography. Since each block stored a hash of its previous block, a single block's modification leads to the re-computation of all the later blocks. Moreover, the decentralized copies of ledgers also significantly increase the difficulties of modification.

2.1.1 Home Chain: Ethereum Mainnet

Ethereum mainnet acts as the root chain. It has many advantages over other blockchains in the market.

Firstly, Ethereum is programmable. Developers are able to write their customized programs, knowing as smart contracts, and deploy them on the chain to offer specific services. Compared to another well-known blockchain, Bitcoin, which focuses more on token transaction and value storage, which only provides Turing-incomplete stack operations, Ethereum gives people a chance to build a multifunctional application.

Secondly, Ethereum has a sizeable token storage and transaction volume. On the one hand, ETH, the native token of Ethereum, is the second most valued and well-known cryptocurrency behind Bitcoin [5], which offers a more significant potential capital entry (see Figure 4). Moreover, Ethereum mainnet has a large variety of ERC20 crypto-tokens on it. Those tokens all follow the ERC-20 Standard [6] and implements standard API, lowering the cost of managing such tokens. The large variety of tokens offers gamers more payment options to buy Gcoins.

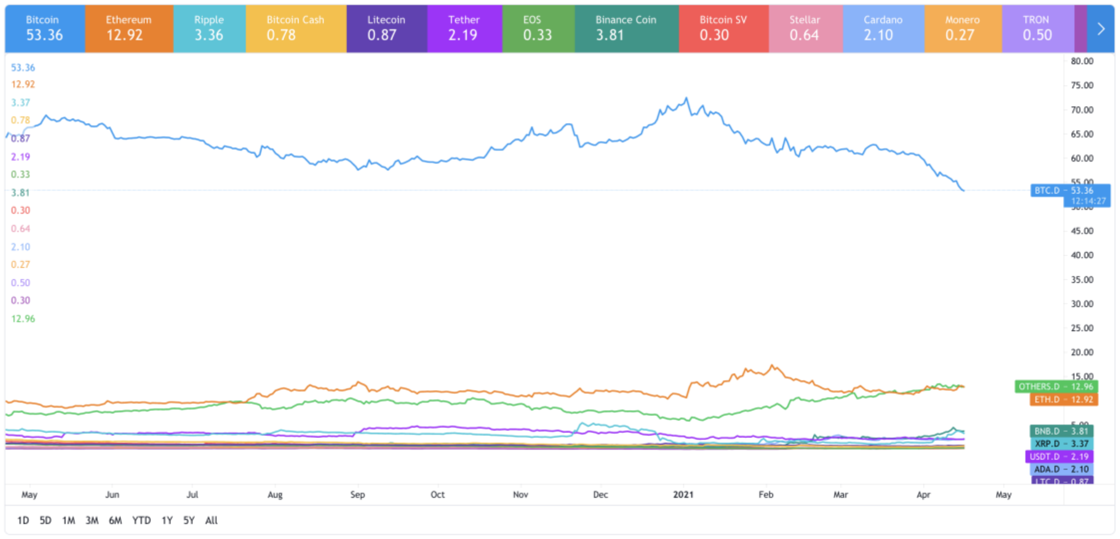


Figure 4. Market capitalization dominance of the cryptocurrency market [5]

Among the ERC20 tokens, there are many stablecoins, such as USDT and USDC. The stablecoins are designed to minimize their price volatility relative to some stable assets. Maintaining a fixed exchange rate with them helps the platform stabilize the price of Gcoin, making it easier to buy Gcoins by e-payment.

In the development of the Gcoin Platform, several smart contracts are deployed on the home-chain to create the platform's entry points.

2.1.2 Objectives for Child Chain

Since Ethereum mainnet is a public chain with nodes worldwide and currently using PoW as its consensus algorithm, the throughput is often low (see Figure 5). Besides, there are usually a large number of transactions (see Figure 6), resulting in a rat race in transaction fees.

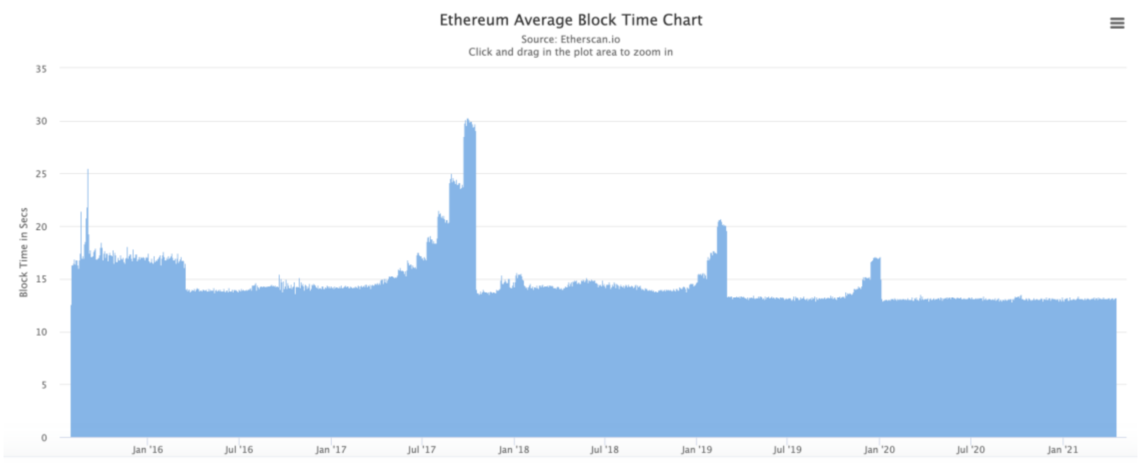


Figure 5. Ethereum average block time [7]

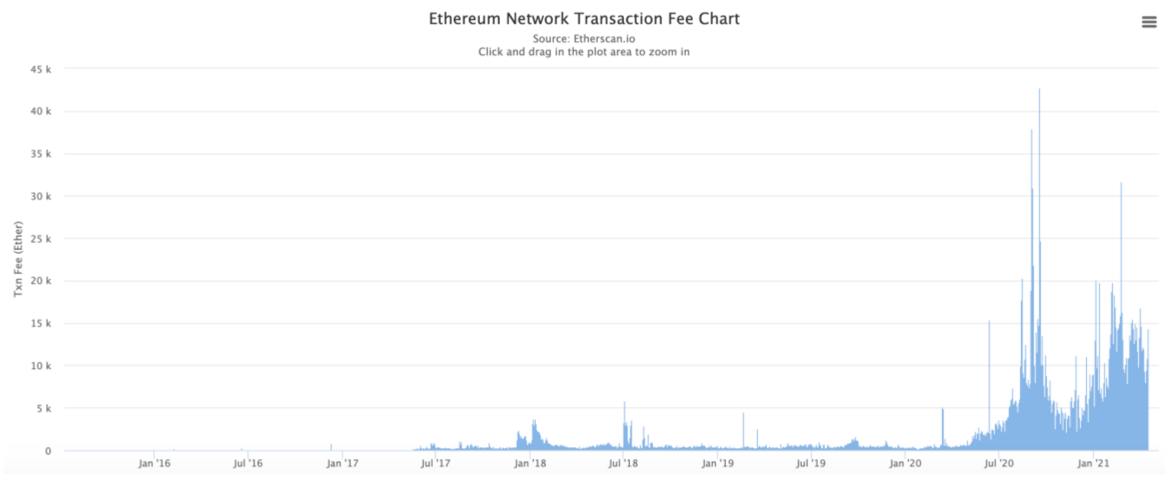


Figure 6. Ethereum transaction fee [8]

All those features have a terrible effect on the platform business, as most gaming assets purchasing are frequent micropayments. The public chain also lacks customization options, and since an arbitrary node can join the network, there is a lack of safety and data privacy.

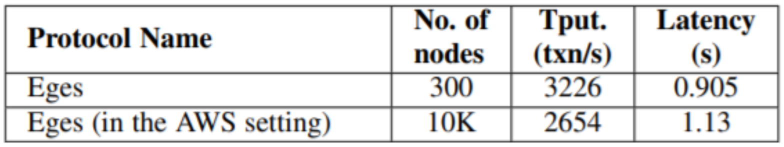
Therefore, a customizable permissioned chain is needed. A cross-chain protocol is used to achieve the token exchange between Ethereum and the permissioned chain.

2.1.3 Child Chain: EGES Permissioned Chain

Rather than deploying all smart contracts and performing all transactions on the mainnet, a permissioned network with entrance on Ethereum mainnet is taken advantage of. The child chain consists of fewer nodes and adopts a faster consensus algorithm, as a result, a smaller the transaction latency is achieved.

The child chain is implemented based on the Ethereum network. As a global and open-source platform, Ethereum is heavily tested and has an active community. As discussed before, it is programmable and allows customized programs containing our business logic. It also offers ERC-20 Standard, s that the platform Gcoin can be released and used by the standard.

A special consensus algorithm called EGES (stands for Efficient, GEneral, and Scalable consensus) is plugged in. It is an algorithm designed specifically for the permissioned chain and is implemented based on Intel SGX, a trusted hardware feature on commodity CPUs [9]. The hardware feature provides a secure environment to hide data and code execution from being seen or tampered with from outside. As a result, it keeps a high consensus efficiency (see Figure 7) while solving the long-discussed problem that permissioned chains are easily targeted for denial-of-service (DoS) attacks [9].



Protocol Name	**No. of nodes**	**Tput. (txn/s)**	**Latency (s)**
Eges	300	3226	0.905
Eges (in the AWS setting)	10K	2654	1.13

Figure 7. EGES is a fast consensus algorithm [9]

2.2 Cross-chain Strategy

A cross-chain strategy is needed to achieve atomic swapping between Gcoin on the permissioned chain and other coins on the Ethereum. Most of the time, the exchange will happen between Gcoin and the stable coin USDT.

2.3.1 Cross-chain Scenarios

In the application, there are two scenarios where stablecoins and Gcoins need to be exchanged. This includes players topping up Gcoin wallets, and gaming companies withdrawing game revenues into stablecoins.

2.3.2 Oracle Service for EVM Obtaining Outside World Data

A significant difficulty in cross-chain transaction is that EVM cannot obtain word data from the internet during execution. Therefore, it could only depend on data submission from external nodes to obtain information in another chain. Oracle is one of the services that help EVM reaching off-chain data. Rather than making requests to the web, oracle nodes will write the data on the chain.

The Oracle service's operation would involve EVM and the oracle nodes (see Figure 8). The oracle service would be run in the following flows for data obtaining:

1. Users make the writing request to the USER Smart Contract
2. User smart contract call Oracle smart contract to use its service for web data obtaining
3. The Oracle contract will emit the event to notify the oracle nodes to do the data request job.
4. Oracle nodes will listen to the events and call the web module, which obtains the data responses from the web
5. Data retrieved will be written to the Oracle smart contract.
6. Based on the voting mechanism, the data value written most will be passed to the User smart contract.

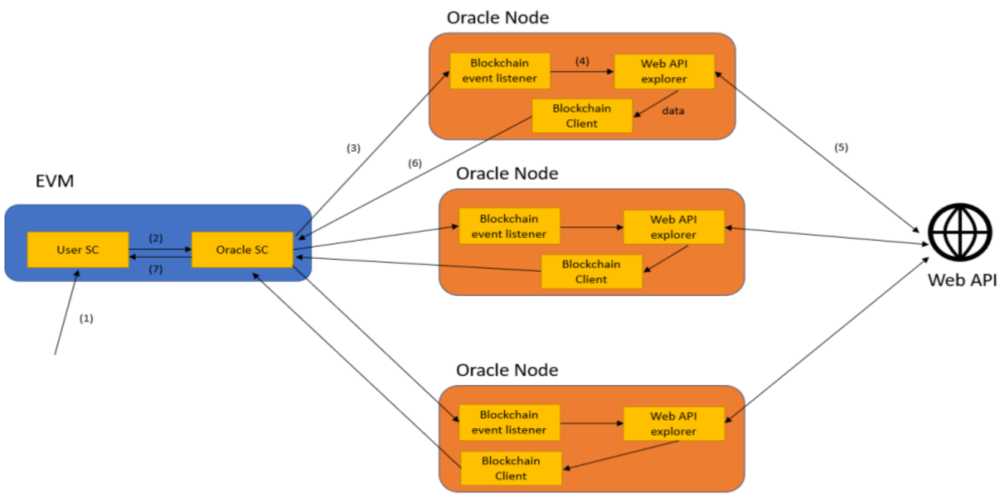


Figure 8. Oracle service's principle

2.3.3 Oracle Cross Chain Protocol

Oracle service acts as a bridge between the Ethereum public chain and the permissioned chain, so that they can look up the data in another chain (see Figure 9). The blockchains become the data-source, and the web module will query for chain data via RPC over http protocol. As another chain's transaction records become accessible, the smart contract could verify the assets and do the swapping across chains.

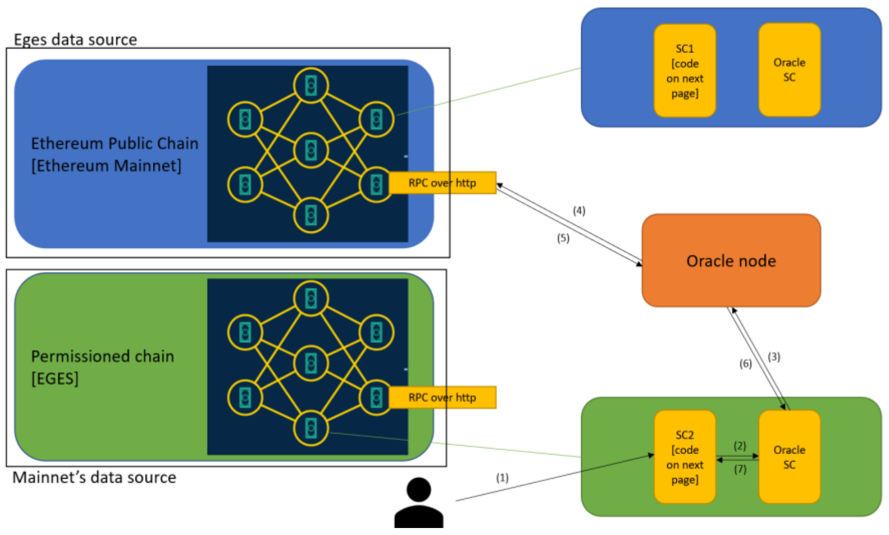


Figure 9. Oracle service for blockchain record obtaining

The bridgable tokens have a special effect in the cross-chain process. It is a pair of tokens conforming both ERC-20 protocol and mining protocols. The tokens are deployed in the home chain and the child chain respectively, and they act like homogeneous tokens between the chains. Whenever a token is transferred to an account, the same account on the other chain should also get one, i.e. bridge contract mines one for him [10]. The implementation principle is listed below:

- On the home chain:
    1. After the target token is transferred to the bridgable contract, the token is locked for the sender's account and an event is emitted.
    2. The Oracle nodes listen for the event and pass the information to the bridgable contract on the child chain
    3. The bridgable contract on the child chain mines a new one and transfer to the sender address on the child chain
- On the child chain:
    1. After the target token is transferred to the bridgable contract, the token is burned and an event is emitted.
    2. The Oracle nodes listen for the event and pass the information to the bridgable contract on the home chain
    3. The bridgable contract on the home chain unlock the corresponding number of tokens and transfer to the sender address on the child chain

To ensure the synchronization of the brigable tokens on both chains, only the brigable contract on the child chain are granted right to do mining. In addition, the user account on both chains should have the same public address.

A withdrawal restriction is applied here. Since only brigable tokens are able to perform cross-chain transaction, a constraint is added on the token exchange process that only the registered account can exchange coins to brigable tokens on the child chain.

In total 6 smart contracts are involved in the process. On the home chain, there are a stablecoin contract, a bridable contract and an exchange contract to exchange the stablecoins with the bridegable coins. On the child chain, there are also three contracts, including Gcoin contract, a bridable contract and an exchange contract to exchange Gcoin with the bridegable coins. Here is an example of topping up:

1. User allows the exchange contract to spend their stable coins
2. The exchange contract exchanges their stable coins to the brigable coins
3. Invoke the cross-chain process in the brigable tokens. If everything goes as expected, the user will get the same number of brigable tokens on the child chain.
4. User allows the exchange contract to spend their brigable tokens
5. The exchange contract exchanges the bbrigable tokens with Gcoin

2.3 Restricted Peer Adding Policy

With concerns about the child chain's efficiency and security, the gaming platform has to suppress the node number's growth speed. A private network is used to form a sealed environment to prevent the invader nodes joining the network from the outside internet.

2.4.1 Limited Nodes Policy

The policy of limited blockchain nodes for each gaming company is adopted to reduce the possibility that a single company being "the majority", who is able to control the whole chain and cause the chain to be mutable. Besides, as the number of nodes in the chain increases, the efficiency of the chain will decrease, as it takes time for nodes to reach agreement and to spread the messages.

2.4.2 Private Network for Access Restriction

All nodes in the child chain should be put in a single private network, which has its own private IP address space. With a well-configured gateway, nodes set in the private network cannot be connected by the unauthorized nodes who want to join the chain. Several special nodes with their public I.P. address assigned will handle the user requests from the internet. To prevent unauthorized node registration, the public-accessible nodes' API of "addPeer()", the function to call when a node want to join in a blockchain network, will be switched off.

2.4 System Design

The platform system is divided into four-level: a well-designed front-end, a back-end system to handle business logic, Ethereum as the home chain, and the permissioned child chain. (see Figure 10).

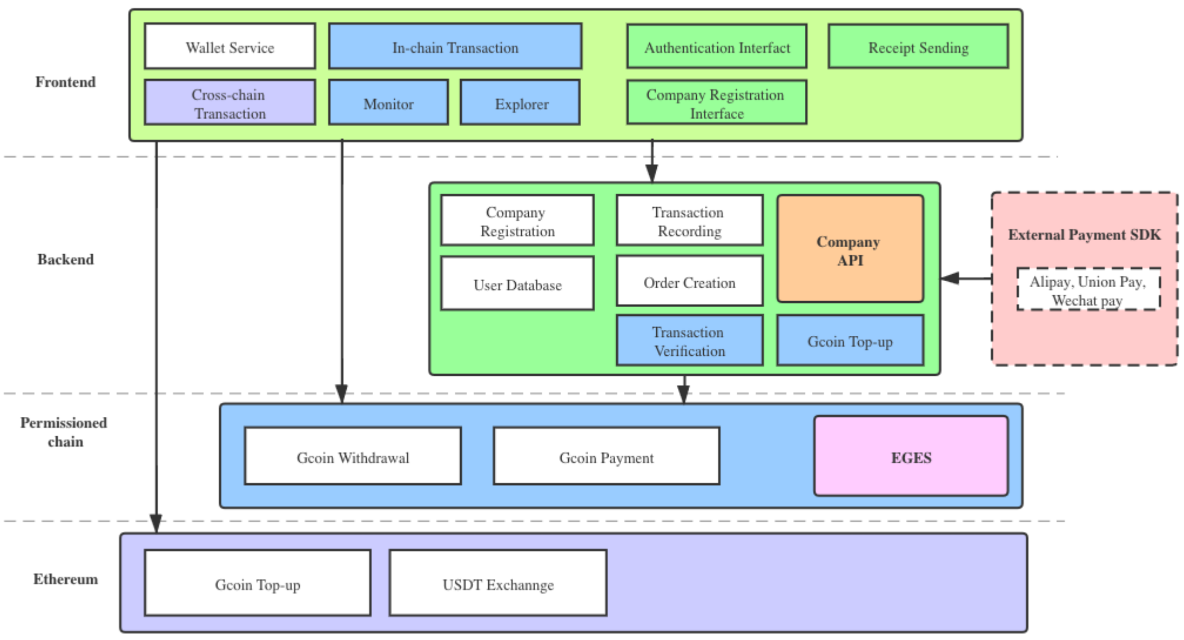


Figure 10. The 4-layered architecture of the Gcoin Platform system

In the home chain, i.e. Ethereum, only smart contracts that are the entry points to the child chain are deployed. These smart contracts, including a smart contract for USDT exchange and a smart contract for the exchange token, are used for user top-up and company withdrawal.

Corresponding cross-chain smart contracts are also deployed in the permissioned chain. Business-related contracts, for example, the contract releasing Gcoin, is also deployed. Instead of the well-known PoW or PoS consensus used in Ethereum mainnet, the child chain adopts EGES consensus, increasing its safety and efficiency.

A front-end is a desktop application written in Vue and wrapped in Electron. It is able to interact with both root and child chains and all transactions are sent directly from the front-end to the chain to keep the application decentralized. A child-chain explorer is integrated to visualize the transactions on the chain, and a monitor is provided for real-time child chain situation monitoring. In addition, a wallet service will be embedded so that users do not need to spend much time managing their account.

The back-end is written in Spring Boot. It handles business work, such as authentication and company registration. MySQL is used to maintain users' and games' information. An API bundle is provided for companies to connect to the platform and to add their own constraints on assets trading. An external payment SDK can be plugged in, so that users can pay the Gcoin Platform directly by e-payment, who then transfer the corresponding amount of Gcoin to their accounts.

The game data server is written in the google firebase cloud function framework using node-js. It deals with all the services related to the in-game trading, including the in-game purchases and trading between players. The trading orders are stored in a NoSQL database. Each time when the server receive a new transaction, it will verify it, check whether it is already in the database to prevent double claiming of the assets, and once the verified transaction is confirmed on the blockchain, it will transfer the assets to the buyer.

3 Implementation

This section will introduce the implementation details of the Gcoin Platform. It is mainly focused on the implementation of cross-chain strategies, Gcoin issuing and the chain interaction, which are done by the author of this paper. Tbe section is divided into the technical and applicational implementation.

3.1 Cross-chain Strategy Implementation

Several cross-chain proposals have been tried during the cross-chain implementation. Most of them have different defects and cannot satisfy the special needs of the Gcoin Platform. The only one strategy that fulfill all the requirements are chosen as the final cross-chain algorithm.

3.1.1 Cross-chain Trials

In total 4 attempts have been practiced. This section will introduce their basic principles and their defects that cause them rejected.

3.1.1.1 Trial1: Intermediary-based Algorithm

This algorithm exchanges coins between different blockchains through two intermediaries. Since Ethereum is programmable, the whole process relies on the smart contracts deployed on Ethereum. Here is an example: Alice want to exchange her Litecoin on Litecoin Blockchain with Bob who has some Bitcoin on Bitcoin Blockchain. She first finds an intermediary C1 who has both account on Litecoin Blockchain and Ethers on Ethereum. The exchangers also need to find another intermediary C2 who has enough amount of Bitcoin and an account on Ethereum. The trading starts as below (see Figure 11):

1. Alice and Bob want to have a deal, using x Litecoin to exchange y Bitcoin
2. The transaction is published, and intermediaries C1 and C2 are decided
3. Alice transfers x Litecoin to C1 on Litecoin Blockchain
4. C2 transfers y Bitcoin to Bob
5. C2 gets ethers of C1 from the pre-deployed smart contracts.

Before the process starts, the intermediaries C1 and C2 should deposit enough ethers, including the amount needed in the transaction and the guarantee deposit. A validation committee who also deposit ethers in the smart contracts will validate the transactions from Alice to C1 and from C2 to Bob before the intermediaries can get ethers from smart contracts. The intermediaries and the validators will get awards after the cross-chain transaction finishes successfully

[11].

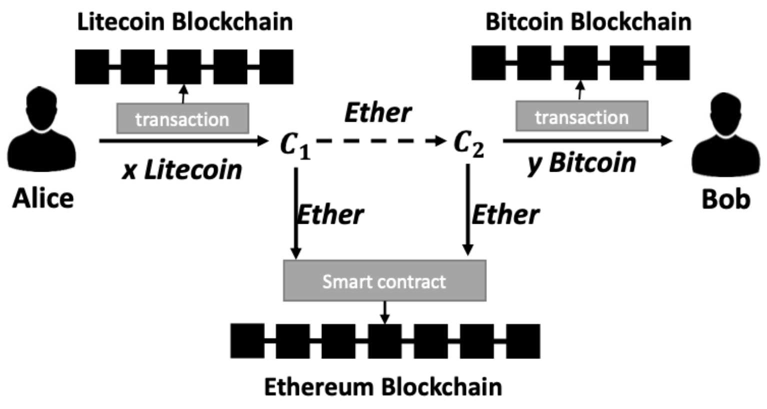


Figure 11. The schema of the intermediary-based cross-chain algorithm [11]

This proposal is plausible, but two defects cannot be ignored. First, the transaction process highly depends on the intermediaries and the validators. Only when there is large enough amount of those entities can the algorithm be reliable. However, this is hard to reach as Gcoin is not a real valued token but only a game coin. Second, rewards must be paid to the intermediaries and the validators each time the cross-chain transaction happens. Since users' deposit may be micropayment transactions, the fees significantly decrease the platform's and companies' revenue.

3.1.1.2 Trial2: DeXTT Algorithm

This algorithm tries to achieve eventual consistency across several chains with the help of contestants, who monitor all chains and spread the newly created transactions among the ecosystem of the blockchains. The process is shown below (see Figure 12):

1. Sender creates an intent (sender, receiver, amount, validity period) and sign.
2. Receiver counter-signs the intent as a Proof of Intent
3. Receiver posts the PoI to a single blockchain in the ecosystem as a CLAIM transaction.
4. Contestants sign the published PoI and propagate it to all blockchains as a CONTEST transaction.
5. After the validity period ends, no new CONTEST transaction can be sent.
6. A FINALIZE transaction is posted. A specific witness is awarded.

The transactions spreading can be done by multiple witnesses in concurrency. However, only the winning witness, i.e. the contestants with a signature closest to zero, is awarded.

The algorithm prevents cross-chain double spending. Any party can post a VETO transaction when two conflicting PoIs are discovered. A contest is also hold and the winner will be awarded after the validity period [12].

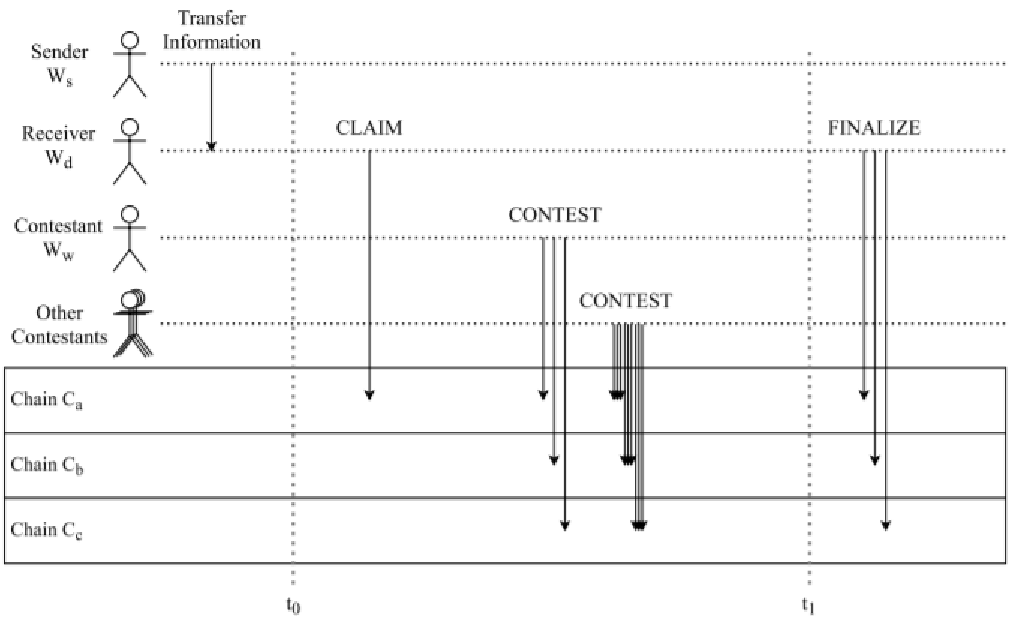


Figure 12. The schema of DeXTT cross-chain algorithm

DeXTT fills the gap of high reward fees from the previously discussed intermediary-based algorithm. However, it is a peer-to-peer cross-chain solution. It requires the signature of both sender and receiver to create the PoI. In contracts, the gaming platform uses smart contracts rather than a real account to perform the cross-chain transaction for decentralization purposes, i.e., the goal is to ensure companies' withdrawal even without the platform's permission. As smart contracts cannot have private key, there is no way for them to sign a transaction and start the chain synchronization.

3.1.1.3 Trial3: Micropayment Channel

Since most of the platform's transactions are micropayments, it is natural to think about whether micropayment channel, an off-chain payment way, can be applied. A micro-payment channel can significantly increase the transaction speed while lowering the transaction fees (see Figure 13).

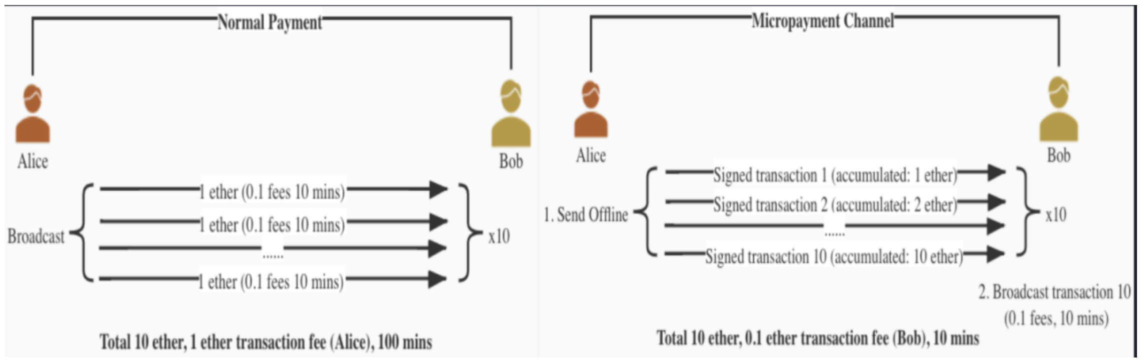

Normal Payment
Alice
Bob
Broadcast
1 ether (0.1 fees 10 mins)
1 ether (0.1 fees 10 mins)
......
1 ether (0.1 fees 10 mins)
x10
Total 10 ether, 1 ether transaction fee (Alice), 100 mins
Micropayment Channel
Alice
Bob
1. Send Offline
Signed transaction 1 (accumulated: 1 ether)
Signed transaction 2 (accumulated: 2 ether)
......
Signed transaction 10 (accumulated: 10 ether)
x10
2. Broadcast transaction 10 (0.1 fees, 10 mins)
Total 10 ether, 0.1 ether transaction fee (Bob), 10 mins

Figure 13. Comparison between normal payments and payment-channel payments

Below is how micropayment works:

1. Alice deploys a smart contract on the blockchain with enough amount of Ethers attached to it.
2. Alice creates a message (recipient, amount, contractAddr, other protection against repay attacks), signs and sends it to Bob
3. Bob send the signed messages to the smart contract and claims the payment.

After the signed message is received, the smart contracts will use the plain information given to reconstruct the message. A Solidity built-in function ‘erecover()’ will be called to produce the public address of the signer from the plain and signed messages. Several methods are implemented to prevent attacks, including the nonce to avoid simple repayment and the contract address against cross-contract repayment [13].

Our trial is to deploy the smart contract on Ethereum and issue the signed transaction on the child chain: Alice deposits Ether on Ethereum, which emits an event of Deposit. The geth clients of the child chain are modified to listen to the Deposit event and transfers Gcoin to users’ account on the child chain. A signed transaction is created and verified by the child-chain smart contracts every time user wants to buy digital assets, and companies can use the signed message to get refund tokens on Ethereum (see Figure 14). The deposit on Ethereum will be locked until someone claims them with the message signed by the owners. To ensure that both chains can verify the signature, users should have the same public address on both chains.

-

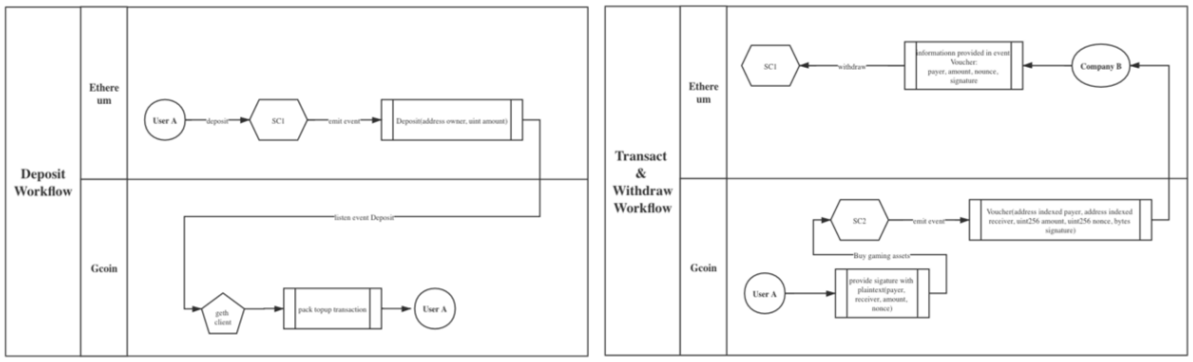

Deposit Workflow
Ethereum
Gcoin
User A
deposit
SC1
emit event
Deposit(address owner, uint amount)
listen event Deposit
geth client
pack topup transaction
User A
Transact & Withdraw Workflow
Ethereum
Gcoin
SC1
withdraw
informationn provided in event Voucher: payer, amount, nounce, signature
Company B
SC2
emit event
Voucher(address indexed payer, address indexed receiver, uint256 amount, uint256 nonce, bytes signature)
Buy gaming assets
User A
provide sigature with plaintext(payer, receiver, amount, nonce)

Figure 14. The workflow of the configured cross-chain payment channel

This method also solves the problem of high reward fees because of the native characteristics of the payment channel. Nevertheless, there is no protection against cross-chain repay attacks. Also, as smart contract cannot sign, the root chain contract is not able to identify whether signed message is issued in the child chain. This gives a chance for the depositor to illegally transfer his deposit away. For example, the payer can sign another message and transfer the deposit to his accomplice before the company request the fund.

3.1.1.4 Trial4: Plasma

Plasma is a famous scaling framework first introduced inn 2017. It is one of the most well-known and practical approach to apply 2nd layer sidechain. Several organizations, such as Omisego, FourthState Lab, etc., are now reaching on it and have some realization.

Plasma deploys several Plasma smart contracts on the root chain to allow users' deposits and withdrawals. Two important states are stored in the Plasma contracts:

- Plasma block list: a list of blocks, each of which stores (1) the Merkle Root (2) block commit time
- Exit list: a list of submitted exit requests, each of which stores (1) the address of the applicant and (2) the location on which the exit request was filed

Several validators periodically submit child-chain status to the home chain to get rewards. When a malicious submission occurs, all users in the sub-chain can submit an anti-counterfeiting proof to the main chain, and the block containing malicious transaction will be rolled back. These block producers lose an on-chain deposit as a result.

This is how user deposits to the child chain. Tokens are sent to the root chain Plasma contract. After some verification, the contract adds the hash of the new block with the single deposit transaction in it to its block list, and then emits an Event for Deposit. The child-chain clients are modified to listen for the event and update user account in child-chain.

The child chain may adopt different consensus algorithms, including PoA. The withdrawal is promised and users can get their assets back to Ethereum anytime they want, even when a single entity has complete control over block production in the sub-chain. Plasma ensures this promise by maintaining an exit list to order the exit request by block heights, and child-chain users can challenge the transaction whenever they found an exit is problematic [14].

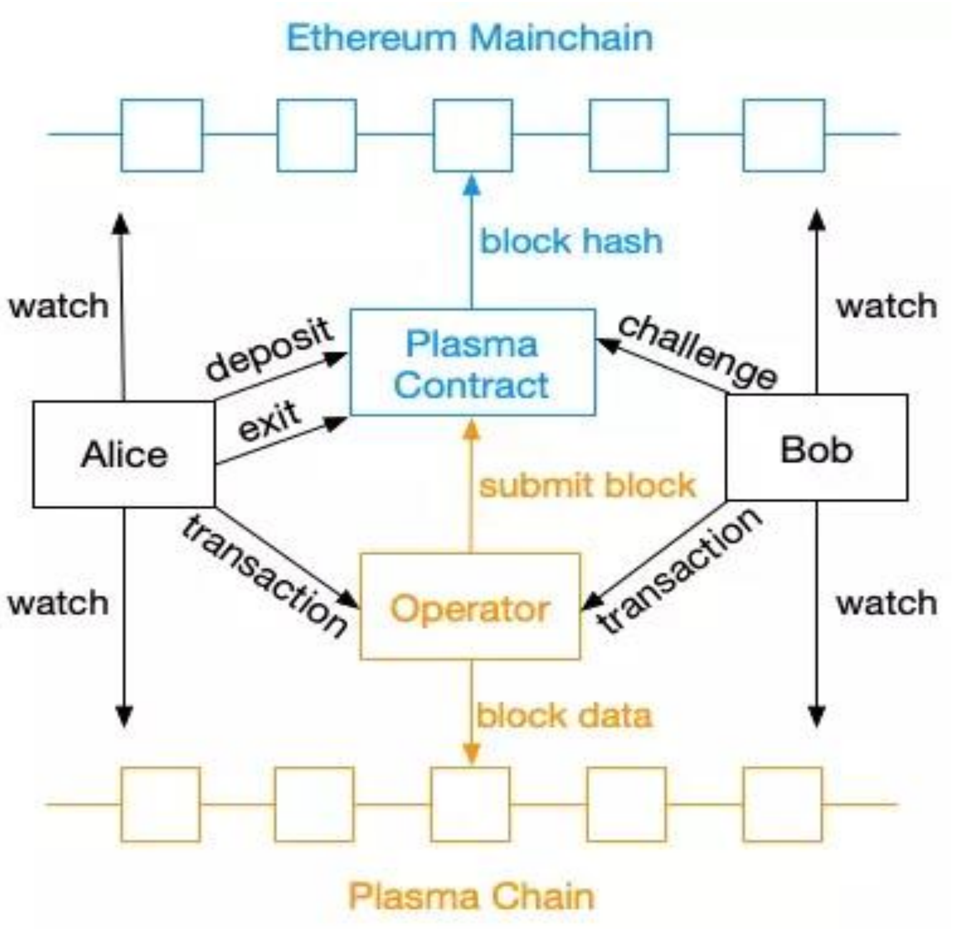


Figure 15. The life circle of Plasma

This algorithm seems to work well. However, limitations cannot be ignored. First, users need to check the home-chain block commitment periodically to ensure the safety of the sub-chain. They also need to pay attention to the exit requests related to their transactions to raise a challenge against malicious exits in time. Second, in order to ensure most users can see and verify the exit requests, the challenge period before the withdrawal becomes valid is usually long, which significantly lowers the cross-chain throughput. Third, the basic concept of Plasma is to promise users leave from the sub-chain. However, since only specific accounts are allowed to take their assets back to the home-chain in the design of the Gcoin Platform, this algorithm cannot fit the platform business scenarios.

3.1.1.5 Conclusion

An acceptable cross chain solution should be satisfactory with 4 use cases.

1. Transactions happens between the smart contract and the peer and should not require the signature of the smart contracts.
2. There are always enough deposits to perform withdrawal.
3. No need for a large number of intermediaries or validators.
4. A permission control of withdrawal is able to be applied.

The above 4 attempts do not fit the platform's business logic well (see Table 1). A new algorithm is needed.

	No need for smart contracts' signature	Valid withdrawal	No need for intermediaries or validators	Permission control is possible
Intermediary-based Algorithm	√	√	**x**	√
DeXTT	**x**	√	**x**	√
Modified payment channel	√	**x**	√	√
Plasma	√	√	√	**x**

Table 1. Cross-chain strategies' features V.S. required use cases

3.1.2 Oracle Cross Chain

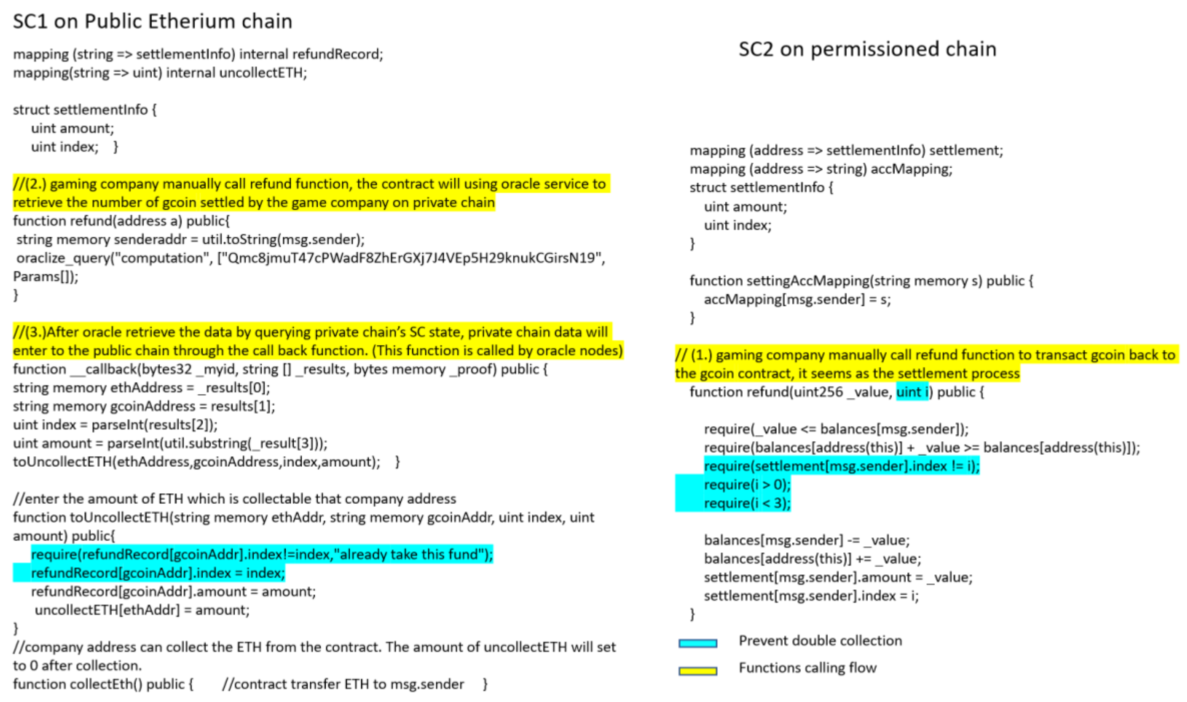
The oracle cross-chain method is the only one solution which is satisfactory for the use cases mentioned before. In this section, USDT withdrawal is used as an example to go through oracle cross-chain implementation on the coding level.

In order for game companies to withdraw stablecoins, the public Ethereum chain relies on a third-party oracle service to find the transaction records of the permissioned chain and verify the settlement of Gcoin. First, the game company will call the refund function of the child chain smart contract to settle the Gcoin obtained from the player. Subsequently, the game company will call the refund function on the smart contract of the public Ethereum chain, and the "oraclize_query" function in the refund function will be triggered to send an event to notify the oracle node that is performing the oracle data verification work. There are 3 parameters passed to the "oraclize_query" function, including the query type, the IPFS hash address of the Dockerfile, and the parameters required by Oracle to verify transaction records. The parameters will take the transaction hash as input.

Through the received IPFS hashed address, the Oracle nodes could pull the Dockerfile which is previously uploaded to the IPFS system. Then the Dockerfile would be built to form an image and run on Provable’s remote oracle nodes to obtain the transaction record corresponding to the transaction hash in the parameter input. Finally, the Oracle nodes would call the _callback function to pass the obtained transaction records from the EGES permissioned chain to the Ethereum public chain to verify caller’s Gcoin settlement transaction.

Through the received IPFS hash address, the Oracle node can pull out the Dockerfile previously uploaded to the IPFS system. It then builds the Dockerfile into an image and run it on a provable remote oracle node to obtain the transaction record corresponding to the transaction hash in the input parameter. Finally, the Oracle node will call the callback function to pass the transaction records obtained from the permission chain to the Ethereum public chain to verify the caller’s Gcoin settlement transaction.

The code highlighted in blue is to prevent game companies from repeatedly collecting stable coins. Double collection means that the account settled Gcoin once, but stablecoins collected on foreign chains more than twice. Updating the transaction index is how the cross-chain solution handles this situation.



SC1 on Public Etherium chain

```
mapping (string => settlementInfo) internal refundRecord;
mapping(string => uint) internal uncollectETH;

struct settlementInfo {
    uint amount;
    uint index;    }

//(2.) gaming company manually call refund function, the contract will using oracle service to
retrieve the number of gcoin settled by the game company on private chain
function refund(address a) public{
 string memory senderaddr = util.toString(msg.sender);
 oraclize_query("computation", ["Qmc8jmuT47cPWadF8ZhErGXj7J4VEp5H29knukCGirsN19",
Params[]);
}

//(3.)After oracle retrieve the data by querying private chain’s SC state, private chain data will
enter to the public chain through the call back function. (This function is called by oracle nodes)
function __callback(bytes32 _myid, string [] _results, bytes memory _proof) public {
string memory ethAddress = _results[0];
string memory gcoinAddress = results[1];
uint index = parseInt(results[2]);
uint amount = parseInt(util.substring(_result[3]));
toUncollectETH(ethAddress,gcoinAddress,index,amount);    }

//enter the amount of ETH which is collectable that company address
function toUncollectETH(string memory ethAddr, string memory gcoinAddr, uint index, uint
amount) public{
    require(refundRecord[gcoinAddr].index!=index,"already take this fund");
    refundRecord[gcoinAddr].index = index;
    refundRecord[gcoinAddr].amount = amount;
     uncollectETH[ethAddr] = amount;
}
//company address can collect the ETH from the contract. The amount of uncollectETH will set
to 0 after collection.
function collectEth() public {       //contract transfer ETH to msg.sender     }
```

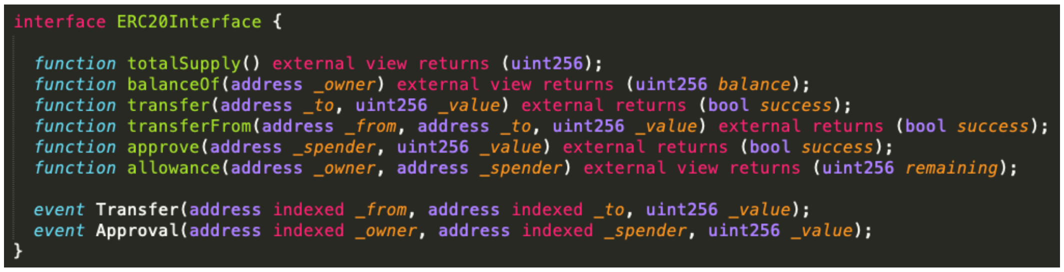
SC2 on permissioned chain

```
  mapping (address => settlementInfo) settlement;
  mapping (address => string) accMapping;
  struct settlementInfo {
     uint amount;
     uint index;
  }

  function settingAccMapping(string memory s) public {
     accMapping[msg.sender] = s;
  }

// (1.) gaming company manually call refund function to transact gcoin back to
the gcoin contract, it seems as the settlement process
  function refund(uint256 _value, uint i) public {

     require(_value <= balances[msg.sender]);
     require(balances[address(this)] + _value >= balances[address(this)]);
     require(settlement[msg.sender].index != i);
     require(i > 0);
     require(i < 3);

     balances[msg.sender] -= _value;
     balances[address(this)] += _value;
     settlement[msg.sender].amount = _value;
     settlement[msg.sender].index = i;
  }
```

Figure 16. Cross-chain smart contract implementation

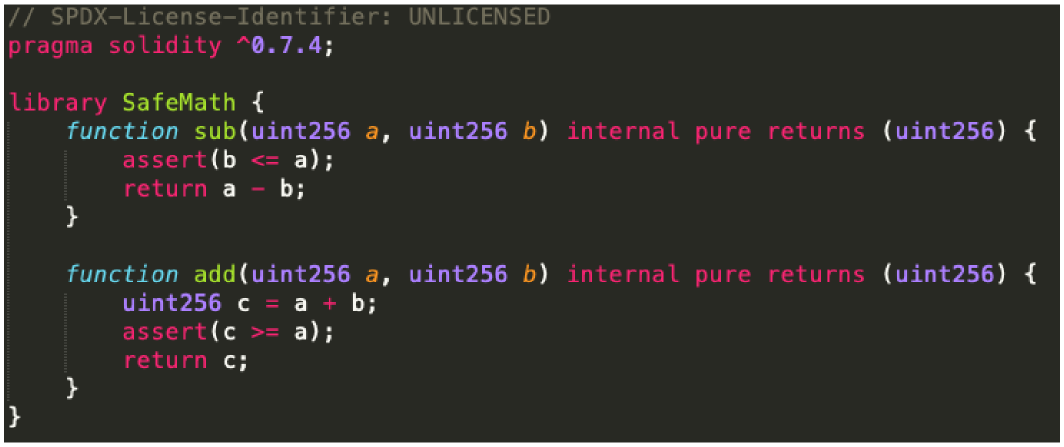
3.2 Gcoin Issuing

The platform’s cryptocurrency Gcoin is issued. It conforms the ERC-20 standard interface introduced in EIP20 [6]. This standard provides basic functionality for token transfers and users can authorize other on-chain third party to spend their tokens (see Figure 17). A simplified version of the library ‘SafeMath’ [15] is used to protect Gcoin contract from overflow and underflow vulnerabilities (see Figure 18).



```
interface ERC20Interface {

  function totalSupply() external view returns (uint256);
  function balanceOf(address _owner) external view returns (uint256 balance);
  function transfer(address _to, uint256 _value) external returns (bool success);
  function transferFrom(address _from, address _to, uint256 _value) external returns (bool success);
  function approve(address _spender, uint256 _value) external returns (bool success);
  function allowance(address _owner, address _spender) external view returns (uint256 remaining);

  event Transfer(address indexed _from, address indexed _to, uint256 _value);
  event Approval(address indexed _owner, address indexed _spender, uint256 _value);
}
```

Figure 17. ERC-20 Interface



```
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.7.4;

library SafeMath {
    function sub(uint256 a, uint256 b) internal pure returns (uint256) {
        assert(b <= a);
        return a - b;
    }

    function add(uint256 a, uint256 b) internal pure returns (uint256) {
        uint256 c = a + b;
        assert(c >= a);
        return c;
    }
}
```

Figure 18. SafeMath applied in Gcoin

Several checks will be performed before real the transfer or allowance action happens. Unlike most of ERC-20 tokens on Ethereum, transfer to zero address is forbidden in Gcoin. This is because the platform tokens are created by making deposits on the root chain, and there is no need to burn tokens to adjust Gcoin's price. In addition, the loss of tokens caused by accidental transfer to zero address can be strictly omitted.

3.3 Blockchain Interaction

This section focuses on the implementation of blockchain interaction, including front-end interaction and backend-interaction. The strategy for wallet service will also be discussed. The section is organized from the front-end to the back-end

3.3.1 Front-end Interaction

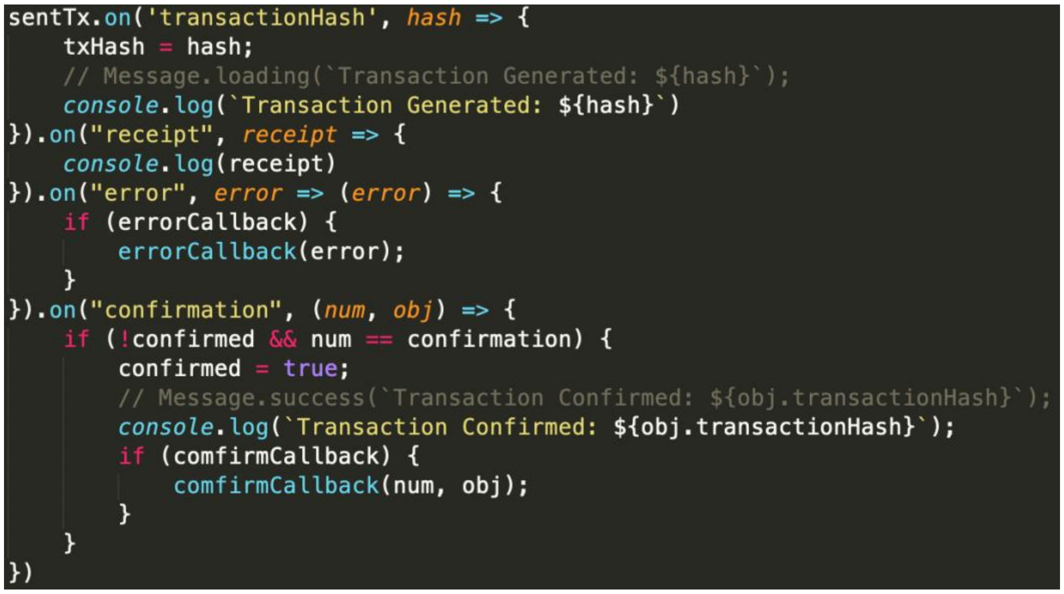
The interaction with blockchains are written in JavaScript and is based on Web3.js, a collection of libraries that enable the interaction with an Ethereum node [16]. Every time when the application is started, web3 instances for the home and child chain will be created to connect to the blockchain. To interact with the deployed smart contracts, the contract addresses and ABIs will be loaded from the config files, and the contract instances and other interactions can be achieved easily with web3 "Contract" API.

All transactions are sent by a global method, which retrieves the user's private key from the local storage, use it to sign the transaction and send it to the corresponding blockchain. In total 4 promise events will be fired (see Figure 19):

1. transactionHash: fired after the transaction sent and returns the hash of the transactions

2. receipt: fired when the transaction receipt is available
3. confirmation: fired when the transaction is included in the blockchain and will be continued fired for the first 12 confirmations. It returns the number of confirmation and the receipt.
4. error: fired if any error occurs when sending the transaction.

A notification will be published for each of the event so that users can clearly know the recent update of their transactions. A call-back function will be called for in-page logic implementation and UI configuration. The decided confirmation times can also be customized.



```
sentTx.on('transactionHash', hash => {
    txHash = hash;
    // Message.loading(`Transaction Generated: ${hash}`);
    console.log(`Transaction Generated: ${hash}`)
}).on("receipt", receipt => {
    console.log(receipt)
}).on("error", error => (error) => {
    if (errorCallback) {
        errorCallback(error);
    }
}).on("confirmation", (num, obj) => {
    if (!confirmed && num == confirmation) {
        confirmed = true;
        // Message.success(`Transaction Confirmed: ${obj.transactionHash}`);
        console.log(`Transaction Confirmed: ${obj.transactionHash}`);
        if (comfirmCallback) {
            comfirmCallback(num, obj);
        }
    }
})
```

Figure 19. Promise events when sending signed transactions

An explorer for the child chain is implemented. It has three search bars, including block searching, transaction searching, and recent N block query. By default, the latest ten blocks will be queried and shown when the page loaded. Each block and transaction can be entered to read its details, including the block height, timestamps, transactions in the block, the miners of the block, etc. and the transaction hash, timestamp, gas used, etc.

A monitor on Ethereum mainnet is also provided (see Figure 20). It uses the external source from ethstats, and users can view the real-time chain status, such as last block commit time, average block time, confirmed blocks, chain's difficulties, etc.

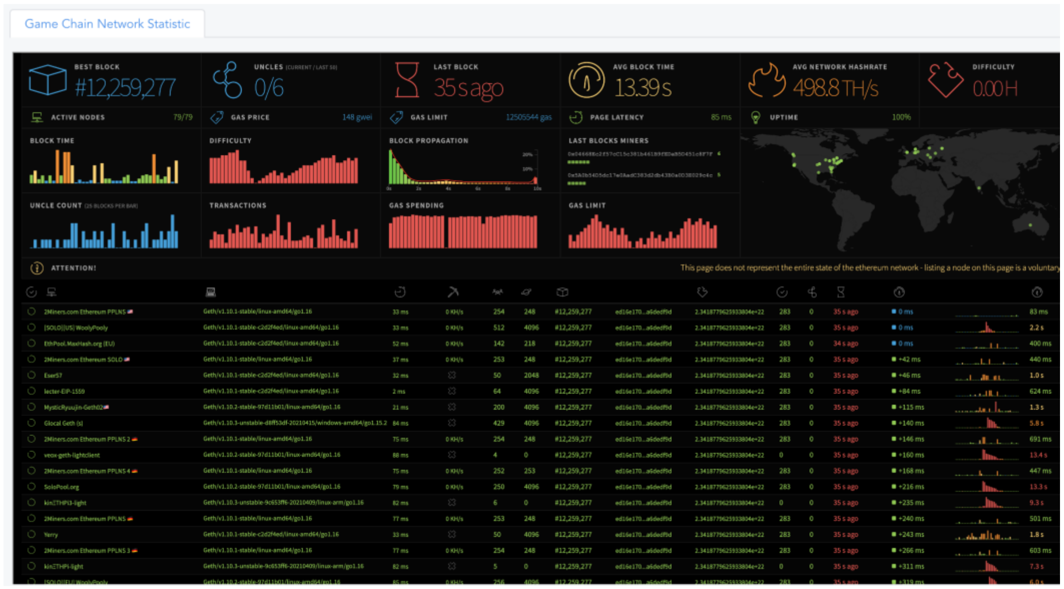
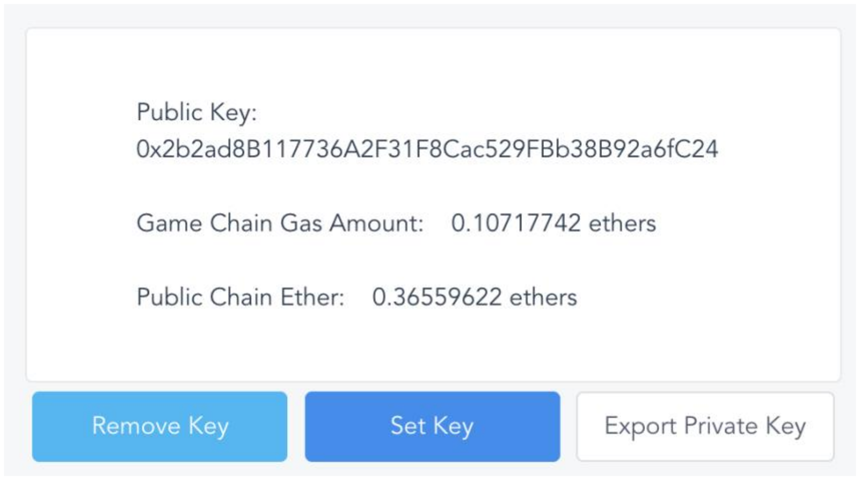


Figure 20. Real-time monitor

3.3.2 Wallet Service

A crypto wallet stores user’s private key and the corresponding public address. The private key will be retrieved locally from the wallet to create the signature whenever a new transaction is initiated. Users could set and clean their keys in the application and their public address will be automatically generated using web3’s “privateKeyToAccount()” API.



Public Key:
0x2b2ad8B117736A2F31F8Cac529FBb38B92a6fC24

Game Chain Gas Amount: 0.10717742 ethers

Public Chain Ether: 0.36559622 ethers

Figure 21. Wallet service management

Since Electron does not support Chrome extension, the famous wallet service MetaMask cannot be used in the application. Since the child-chain accounts are only used in the gaming platform, it is of larger tolerance of wallet safety. Currently, the platform stores users’ private key in their local storage, which is kept in the local computer and will never be involved in any networking event.

3.3.3 Back-end Transaction processing

The in-game purchase process is the core process that integrates blockchain payments with the game's back-end assets transfer. The child chain, the front-end system, the platform server and the game server are involved.

The process goes as below:

1. The front-end system starts a request to buy a specific gaming asset
2. The platform server processes the request and generate an order id
3. Front-end initiates a block-chain payment with order id attached in the extra data field.
4. Front-end submits the transaction hash together with the order id to the platform server.
5. The backend server verifies and validates the transaction.
6. The platform server checks its database for the transaction to avoid double claiming. Then records the transaction in its database
7. The platform backend notifies the game server that the transaction is settled.
8. The game server delivers the item to the gamer.

To prevent a transaction being used to buy different assets, the order id generated by the server must be included in the transaction extra data space and sent to the chain (see Step 3). To prevent players from double claiming an assetsusing the same transaction, the platform data server will refer to its database to see whether this transaction hash is used and stored previously (see Step 6).

3.4 Development Process

The section displays the past development history, including the development stages and development time table.

3.4.1 Development Stages

There are five development stages in total, including the blockchain development, the front-end development, the distributed network deployment, business related development and the final testing and refinement (see Figure 22).

In the first stage, the cross-chain protocol was designed and implemented. The platform's crypto-token was also developed and tested on chain. Then a desktop application front-end was developed in Vue, and the codes related to Web3 written in JavaScript was written so that the blockchains can be directly interacted with from a user-friendly interface. In the third phrase, a customized child-chain network was built using Geth client on AWS. After that, business-related logic, such as company registration, assets payment processing was decided. Back-ends were implemented and put on the server provided Amazon Web Service. Several testing were done to ensure the platform's reliability and efficiency.

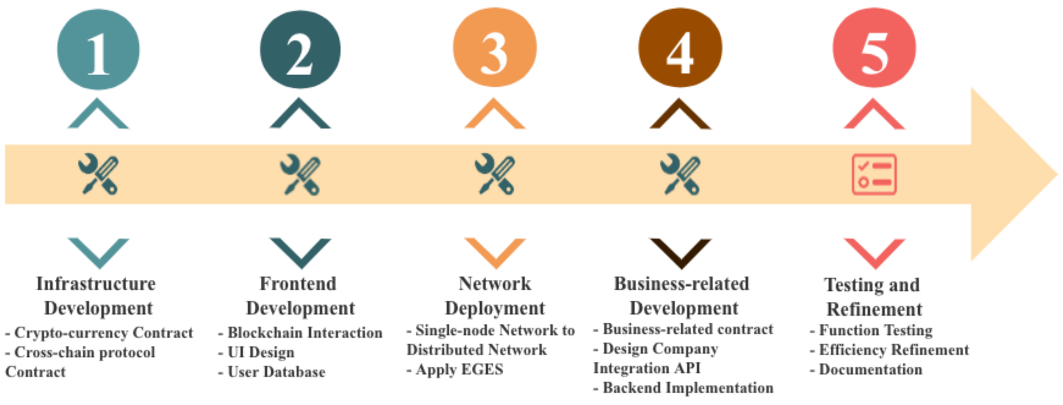

1
2
3
4
5
Infrastructure Development
- Crypto-currency Contract
- Cross-chain protocol Contract
Frontend Development
- Blockchain Interaction
- UI Design
- User Database
Network Deployment
- Single-node Network to Distributed Network
- Apply EGES
Business-related Development
- Business-related contract
- Design Company Integration API
- Backend Implementation
Testing and Refinement
- Function Testing
- Efficiency Refinement
- Documentation

Figure 22. Development stages of the Gcoin Platform

3.4.2 Schedule

Time Periods	Tasks
Sep 2020	Literature Review ● Ethereum Codebase ● EGES
	Deliverables of Phase 1 ● ***Detailed Project Plan*** ● ***Project Web Page***
Oct-Dec 2020	Literature Review ● Ethereum smart contract ● Cross-chain transaction protocol
	Project Design ● Cross-chain protocol selection ● Smart Contract design

	● Architecture design ● Database design
	Project Implementation
Dec-Jan 2020	Project Implementation
	First presentation
	Deliverables of Phase 2 ● ***Preliminary implementation*** ● ***Detailed interim report***
Feb 2020	Project Testing ● Functional testing ● System testing ● Performance evaluation
	Project Refinement ● Performance improvement ● Performance re-evaluation
Mar-Apr 2020	Project Documentation • API document • White paper
	Deliverables of Phase 3 ● ***Finalized tested implementation*** ● ***Final report***
	Final Presentation
May 2020	***Project Exhibition***
	Project Competition

Table 2. Development schedule

4 Future Work

The platform has already been completed. However, future improvements can be down to prepare the platform for production mode. This section is divided into the suggested improvement on the cross-chain algorithm and other applicational improvements.

4.1 Cross-chain Algorithm Improvement

The current Oracle cross-chain algorithm still has some limitations, and two improvements are suggested.

4.1.1 Oracle Cross-chain's Limitations

The oracle cross-chain prototype suffered from expensive cross chain transaction cost and centralized data source.

All Oracle nodes act as validators in data validation and each node must write a response from the data source to the chain (see Figure 23). Most of the data values retrieved from the data source will be passed to the user's smart contract via the Oracle contract. As cross-chain provides blockchain status updates equal to the number of nodes in a single cross-chain transaction, a costly transaction cost will be caused. In addition, as the number of Oracle nodes increases, so does the number of transaction costs.

Furthermore, all the Oracle nodes read the contract state with one blockchain client, which is contrary to the principle decentralization. When the blocking node becomes malicious and responds to fraudulent information, the correctness of blockchain data will be destroyed and the blockchain application is no longer protected from fraud. In case the data source provided node is manipulated by a single entity, the application will lose its objective of decentralization to some extent.

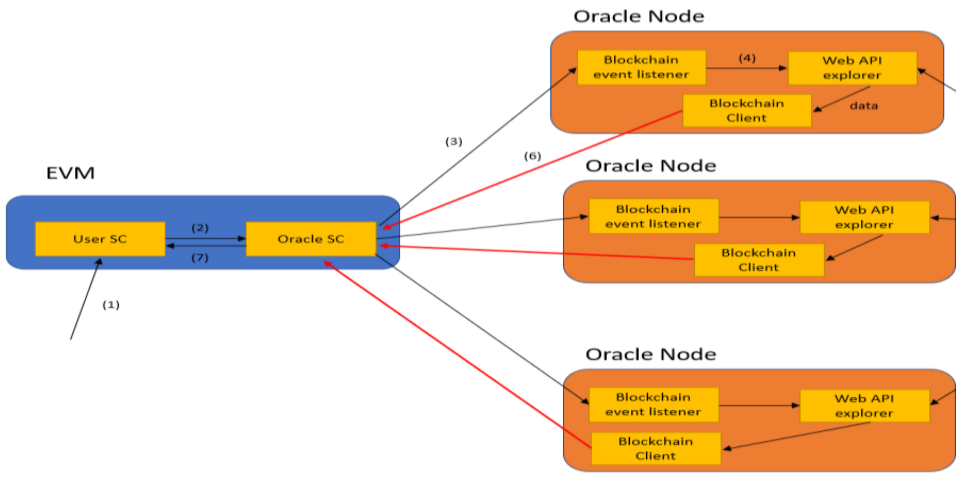


Figure 23. Multiple state updates are required in Oracle cross-chain

4.2.1 Multi Signature

Multi-signature wallet methods are suggested in order to lower cross chain transaction cost. An example is another Oracle cross chain project called token bridge [17].

Multi-signature enables a group of signers to produce a compact, joint signature on a common document or message [18]. Instead of directly writing the foreign chain's data to the Oracle smart contract, the Oracle nodes will sign the transaction with their private key and then send the signed message to the admin oracle node. The admin node waits for enough signatures to be received and then write the signed message to the chain. Since only one state update is applied, the cross-chain transaction cost is largely reduced.

4.2.2 Decentralizing Data-source

The cross-chain implementation currently uses Provable, a third-party Oracle service, with which the oracle nodes obtains the Dockerfile from InterPlanetary File System (IPFS) through the hashed file address [19]. Since all the oracle nodes obtains the same Dockerfile configured with a single remote blockchain client's RPC URL, all nodes then connect to a single client. Decentralized data-sources can be achieved by developing a customized Oracle service that allows nodes to visit arbitrary data-source URL, our oracle service's validation nodes are just required to visit two data-source URLs.

4.2 Applicational Improvement

Three aspects of work will be discussed below, including the safety improvement on wallet service, multi-language supported SDK and the multi-platform application.

4.2.1 Safer Wallet Service

Since current wallet directly store the private key in the local storage, the account is to some extent vulnerable to hacking. Suggested solution is to asymmetrically encrypt the private key before storing it to the local storage. However, a user password is needed every time when a new message needs to be signed.

External wallet service can also be integrated. The suggestion is to either customize Electron to integrate MetaMask in the application, or turn to other reliable services. WalletService is recommended, which is an open source protocol that enable the connection from Dapps to mobile wallets with QR code scanning or deep linking [20]. However, further research on it should be done before the integration.

4.2.2 Multi SDK Programming Language Interface

According to the current development status, JavaScript is the only programming language provided by the in-game payment system SDK. Considered the commonly used game development languages, the development team will also provide C# and C++ language interfaces to lower the barriers of integrating in-game payment systems.

4.2.3 Mobil Application Development

The platform is currently mainly supported as desktop application. However, there is a tendency of moving from desktop and console games to mobile games. The revenues from mobile games grew from $62.3 billions in 2018 to $ 86.3 billions in 2020 and will continue to grow (see Figure 24).

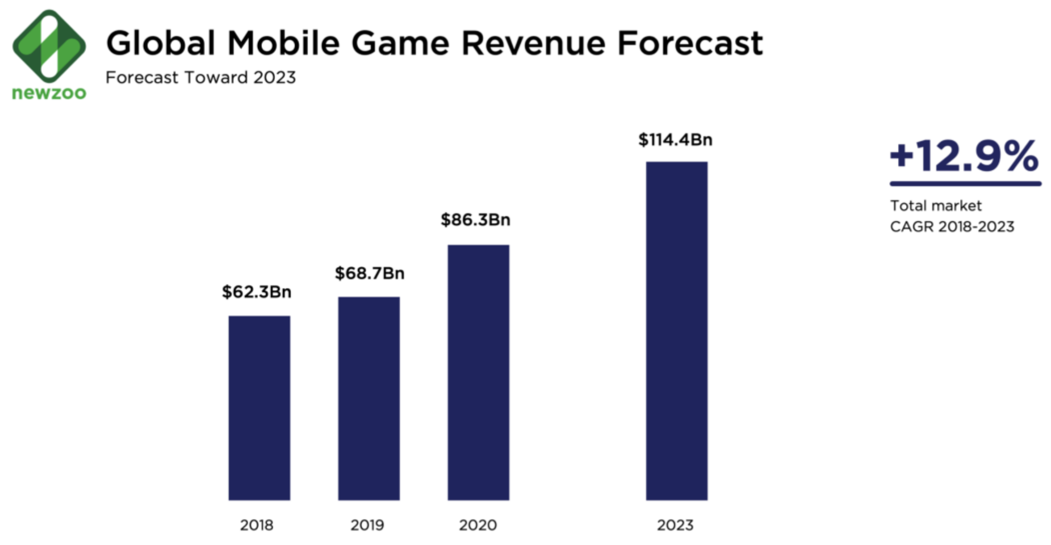


Figure 24. Gobal mobile game revenue

Therefore, mobile apps in Android and iOS are in need, and the in-game payment SDK written in Java and Object-C should also be provided.

5 Conclusion

This paper proposed to build an Etherum-based platform, which issues cryptocurrency Gcoin and handles all game assets trading. A detailed transaction model is provided and until today, the platform has provided five main services, including the wallet service, in-game transaction, cross-chain top-up and withdrawal, game issuing and blockchain monitoring.

To improve efficiency and increase usability, the platform sets up a permissioned chain to handle business transactions and only deployed child-chain entry points on Ethereum, the public chain. Several cross-chain algorithms are tried, such as the intermediary-based algorithm, DeXTT, modified payment channel, Plasma and Oracle cross-chain strategy. The first four trials all have limitations that contracts with the platform's principle and Oracle is chosen. To join the permissioned network, game companies should provide their own nodes together with an Oracle node that involves in the cross-chain deposit and withdrawal.

The platform is divided into four levels, including the home chain, the child chain, several back-end servers to handle the assets selling business, and a front end provided as desktop application that interacts with all other three layers. The back-end servers have been already deployed to the servers provided by AWS.

Several future improvements are suggested, including the improvement on Oracle cross-chain algorithm, the wallet service, the payment SDK and mobile applications.

The Gcoin Platform has solved many problems in the gaming industry, such as its intransparency, long periods of revenue withdrawal, etc. Since there is lack of a platform which union different games and offers tokens that can be used across the games, we hope our project could give a hand to the ecosystem of the virtual world of the games。

REFERENCES

[1] D. Takahashi, "SuperData: Games grew 12% to $139.9 billion in 2020 amid pandemic," SuperData, 6 Jan. 2021. [Online]. Available: https://venturebeat.com/2021/01/06/superdata-games-grew-12-to-139-9-billion-in-2020-amid-pandemic/. [Accessed 10 Apr. 2021].

[2] NewZoo, "2020 Global Games Market Report (Free Version)," 2021. [Online]. Available: https://platform.newzoo.com/resources/trend-reports/free-global-games/free. [Accessed 11 Apr. 2021].

[3] S. Direct, "Joining The Steamworks Distribution Program," Steam, [Online]. Available: https://partner.steamgames.com/steamdirect. [Accessed 5 Oct. 2020].

[4] "How much does Apple charge for free apps hosted on the app store?," Apple, 2018. [Online]. Available: https://developer.apple.com/forums/thread/76587. [Accessed 1 Mar. 2021].

[5] TradingView, "Cryptocurrency market," [Online]. Available: https://www.tradingview.com/markets/cryptocurrencies/global-charts/. [Accessed 16 Apr. 2021].

[6] E. I. Proposals, "EIP-20: ERC-20 Token Standard," [Online]. Available: https://eips.ethereum.org/EIPS/eip-20. [Accessed 22 Oct. 2020].

[7] Ingo Weber, Vincent Gramoli, Alex Ponomarev, Mark Staples, Ralph Holz, An Binh Tran, Paul Rimba, "Ethereum average block time chart," Etherscan, [Online]. Available: https://etherscan.io/chart/blocktime. [Accessed 17 Apr. 2021].

[8] "Ethereum Network Transaction Fee Chart," [Online]. Available: https://etherscan.io/chart/transactionfee. [Accessed 17 Apr. 2020].

[9] Xusheng Chen, Shixiong Zhao, Cheng Wang, Haoze Song, Jianyu Jiang, Ji Qi, Tsz On Li, T.-H. Hubert Chan, and Heming Cui, "Efficient, DoS-resistant Consensus for Permissioned Blockchains," 2018. [Online]. Available: https://hemingcui.github.io/draft-papers/eges-consensus.pdf. [Accessed 20 Sep. 2020].

[10] "Token Bridge," [Online]. Available: https://docs.tokenbridge.net/. [Accessed 10 Jan. 2021].

[11] Hangyu Tian, Kaiping Xue, Shaohua Li, Jie Xu, Jianqing Liu, Jun Zhao, "Enabling Cross-chain Transactions: A Decentralized Cryptocurrency Exchange Protocol," 7 May. 2020. [Online]. Available: https://arxiv.org/pdf/2005.03199.pdf. [Accessed 17 Aug. 2020].

[12] Michael Borkowski, Marten Sigwart, Philipp Frauenthaler, Taneli Hukkinen, Stefan Schulte, "DeXTT: Deterministic Cross-Blockchain Token Transfers," 12 Aug. 2019. [Online]. Available: https://www.researchgate.net/publication/335133601_DeXTT_Deterministic_Cross-Blockchain_Token_Transfers. [Accessed 21 Nov. 2020].

[13] "Solidity by Example - Micropayment Channel," [Online]. Available: https://docs.soliditylang.org/en/v0.8.0/solidity-by-example.html#micropayment-channel. [Accessed 15 Nov. 2020].

[14] Joseph Poon, Vitalik Buterin, "Plasma: Scalable Autonomous Smart Contracts," 11 Aug. 2017. [Online]. Available: https://plasma.io/plasma.pdf. [Accessed 27 Dec. 2020].

[15] OpenZeppelin, "SafeMath," 18 Nov. 2020. [Online]. Available: https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/math/SafeMath.sol. [Accessed 20 Nov 2020].

[16] "Ethereum JavaScript API," [Online]. Available: https://web3js.readthedocs.io/en/v1.3.0/. [Accessed 10 Nov. 2020].

[17] Oracle, "Token Bridge Docs," [Online]. Available: https://docs.tokenbridge.net/ . [Accessed 24 Oct. 2020].

[18] Duc-Phong Le, Guomin Yang, Ali Ghorbani, "A New Multisignature Scheme with Public Key Aggregation for Blockchain," 6 Jan 2020. [Online]. Available: https://ieeexplore.ieee.org/document/8949046. [Accessed 25 Nov. 2020].

[19] "Provable Documentation," 28 Aug. 2019. [Online]. Available: https://docs.provable.xyz/#home. [Accessed 12 Oct. 2020].

[20] "WalletConnect - Open protocol for connecting Wallets to Dapps," [Online]. Available: https://walletconnect.org/. [Accessed 27 Jan. 2020].