Article

Non-Face-to-Face P2P (peer-to-peer) Real-time Token Payment Blockchain System

Hyug-Jun Ko [1], Seong-Soo Han [2]* and, Chang-Sung Jeong [2]*

[1] Affiliation 1; doltwo@korea.ac.kr
[2] Affiliation 2; sshan1@kangwon.ac.kr
[3] Affiliation 3; csjeong@korea.ac.kr

* Correspondence: sshan1@kangwon.ac.kr; Tel.: +82-10-2274-0155

Abstract: With the increase of intelligent voice phishing and the increasing reliance on open banking systems, there has been a rise in cases where individuals' personal information has been exposed, resulting in significant financial losses for the victims. Non-face-to-face transactions in the financial sector face challenges such as customer identification, ensuring transaction integrity, and preventing transaction rejection. Blockchain-based distributed ledgers have been proposed as a solution, but their adoption is limited due to the difficulty of managing private keys and the burden of gas fees management. This paper proposes a non-face-to-face P2P real-time token payment system that minimizes the risk of key loss by storing private keys in a keystore file and database through a server-based key management module. The proposed system simplifies token creation and management through a server-based token management module and implements an automatic gas charging function for smooth token transactions. Transaction integrity and non-repudiation are ensured through a transaction confirmation module that uses transaction IDs without exposing personal information. Furthermore, advanced security measures such as blocking foreign IP access and DDoS defense are implemented to securely protect user data. The proposed system aims to provide a convenient, secure, and accessible online payment solution to the public by implementing a self-authentication function using a web application that is not limited to smart phones or application platforms.

Keywords: Blockchain, Symverse, Ethereum, Payment, gas, voice fishing, fin-tech

1. Introduction

The rapid advancement of technology and widespread adoption of digital financial services have opened up a new era of convenience and efficiency in transactions. However, along with these benefits, the possibility of financial fraud has also increased, and the risk of financial fraud is increasing as the sophistication of voice phishing and the dependence on open banking systems increase [1]. Since these kinds of systems integrate information from various financial institutions, the criminals create potential opportunities to exploit vulnerabilities that may result in significant financial losses to individuals whose personal information is inadvertently exposed [2].

In the area of non-face-to-face transactions, challenges such as customer identity verification, ensuring transaction integrity, and preventing transaction refusal still remain. These obstacles hinder the introduction of safe and smooth financial services that protect users from fraud while maintaining ease of use. Blockchain technology, with its unique characteristics of decentralization, immutability, and transparency, has been proposed as a potential solution to resolve these challenges. However, the widespread adoption of blockchain-based distributed ledgers has been limited by the complexity of private key management and the burden of gas coin management and gas costs incurred by transactions [3, 4].

This paper proposes a new non-face-to-face P2P real-time token payment system that overcomes limitations while leveraging the benefits of blockchain technology. The proposed system effectively minimizes the risk of losing private keys by utilizing a server-based key management module to securely store private keys in keystore files and databases. In addition, a server-based token management module simplifies token generation and management and enables smooth token transactions without the burden of gas cost management by implementing an automatic gas charging function [5].

The system's transaction verification module ensures transaction integrity and non-repudiation using transaction IDs without disclosing personal information. Furthermore, it realizes the protection of user data by use of advanced security measures, such as blocking foreign access IP and implementing DDoS protection. The proposed system aims to provide a convenient and safe online payment solution to the general public by implementing self-verification functions using web applications which restricted by types of smartphones and platforms [6].

In chapter 2, an overview of the challenges faced by remote transactions and existing solutions is examined, and the potential of blockchain technology to solve these challenges is discussed. In chapter 3, we introduce our proposed remote P2P real-time token payment system. Chapter 4 confirms the service availability through performance testing, and chapter 5 evaluates the real-time token payment system.

2. Related works

In this chapter, we examine previous research and solutions proposed to address the issues of remote transactions, with a particular focus on customer identity verification, transaction integrity, and prevention of transaction rejection. We also discuss the potential of blockchain technology in addressing these issues and emphasize the limitations of current blockchain-based solutions [7].

2.1. Customer Identity Verification

As non-face-to-face transactions become more popular, verifying the identity of customers has become an important factor in maintaining the security of financial systems. To address this issue, various technologies such as two-factor authentication (2FA), biometric authentication, and digital identity solutions have been proposed [8]. While these approaches have shown promise, they can be limited in adoption and effectiveness due to the need for additional hardware, software, or user intervention. In this context, using a web application that is not restricted by smartphones or platforms for identity verification can provide a more accessible and user-friendly solution, as proposed in the system described by Twitter [9, 10].

2.2. Transaction Integrity and Non-repudiation

Transaction integrity and non-repudiation capabilities are essential for preventing fraud and maintaining trust in financial systems. To achieve these goals, traditional methods such as digital signatures and cryptographic hashing have been widely used. However, these methods can be resource-intensive and may not be suitable for all remote transaction scenarios. Blockchain technology, which has unique characteristics such as immutability and transparency, is being proposed as an alternative. Smart contracts and consensus algorithms have been developed to enforce transaction rules and maintain system integrity. Nevertheless, the adoption of blockchain-based solutions is limited by issues such as personal key management and gas fee management [11].

2.3. Blockchain-based Distributed Ledger

Blockchain technology is receiving significant attention as a potential solution to various challenges in the financial sector, such as decentralization, security, and transparency. Several blockchain-based platforms, such as Ethereum and Hyperledger, have been developed, enabling the creation and management of decentralized applications, including financial transactions. While these platforms have demonstrated potential, they have been limited in widespread adoption due to the difficulty of managing personal keys and the burden of gas costs. Furthermore, scalability and interoperability issues between different blockchain platforms remain significant challenges [12-14].

2.4. Limitations of Existing Solutions

While existing solutions have addressed some issues with remote transactions, they often fall short in providing comprehensive, user-friendly, and secure access. Blockchain-based solutions are limited in adoption due to the complexity of managing personal keys and the burden of gas costs. Traditional methods for identity verification and transaction integrity enforcement may be resource-intensive and cumbersome [15]. In light of these limitations, the proposed remote P2P real-time token payment system aims to leverage the advantages of blockchain technology and integrate user-friendly features for identity verification, transaction integrity, and non-repudiation, while resolving the issues of personal key management and gas costs to bridge this gap. Additionally, we will examine the background technical details of the system and explore the use of Ethereum's keystore and the SCT and transaction generation of the Symverse blockchain in its actual implementation.

2.5. Background Technology

2.5.1. Ethereum Keystore

Ethereum Keystore is a means of authenticating oneself for a specific Ethereum address, and it is a file that encrypts Private Key with Passphrase [16, 17]. To obtain a Private Key, one must know both the Keystore file and the Passphrase, and for usability purposes, the Private Key is not directly exposed and instead, the Keystore and Passphrase combination is used to create a secure standard for transactions [18].

- Generating a Keystore

The Ethereum platform generates a Private Key and a Public Key using the ECDSA (Elliptic Curve Digital Signature Algorithm). The passphrase is encrypted using a one-way cryptographic algorithm called "Scrypt" to generate a Derived Key, as shown in Fig. 1 [19].

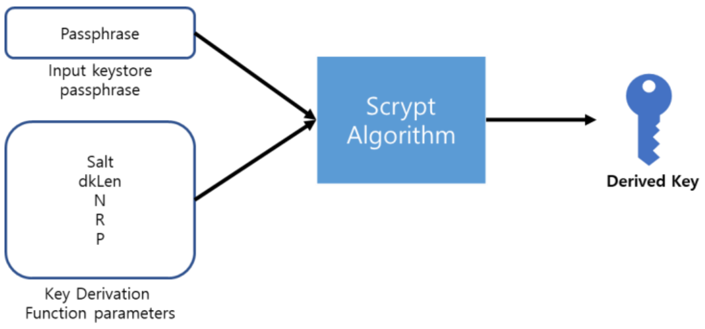


Figure 1. Generate a derived key

For the decryption of the Private Key, it is encrypted using the AES algorithm as shown in Fig. 2 and then it needs the generation of Cipher Text.

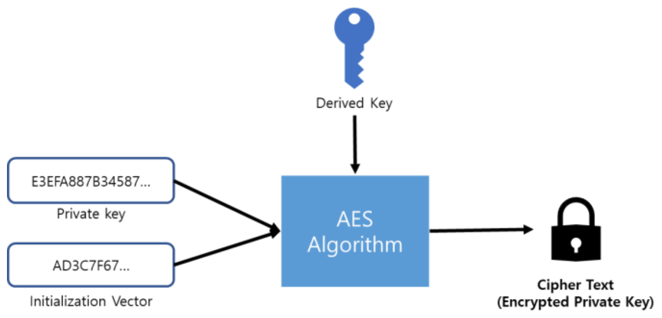


Figure 2. Create a cipher text

The MAC for verifying whether the user-input Passphrase matches is stored in the keystore by concatenating the last 16 bytes of the Derived Key (32 Bytes) with the Cipher Text, and hashing the result using the SHA3-256 hash function, as shown in Fig. 3.

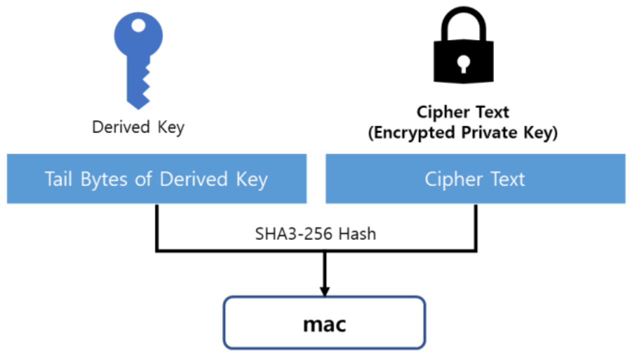


Figure 3. Create mac from derived key and cipher text

The resulting keystore file created in this way is shown in Fig.4.

```
{
   "version":4,
   "id":"64cbaeb8-431b-41d4-a5e6-0508fc509b74",
   "address":"0x1bff0b319a73b51159fb4e2d0111d5c93fa1b3d6",
   "crypto":{
      "ciphertext":"1ab011aeae5d288465a1f4c89cf6b4a494ba90d908ef015ffa43fa9838ff1483",
      "cipherparams":{
         "iv":"2dd0641c64f19d978854a0ab3e27c0a8"},
         "cipher":"aes-128-ctr",
         "kdf":"scrypt",
         "kdfparams":{
            "dklen":32,
            "salt":"cfaeccc4d27f0305f0af2d3d87214a360a06714cdf2320db44c815b9c03b4ce8",
            "n":4096,
            "r":8,
            "p":1
            },
         "mac":"5ad80b19d7338245fb12129c2c441eee104ab054171edfd07c44cd602cdefdf6",
         "machash":"sha3256"
   }
}
```

Figure 4. Keystore file example

- Decrypting Keystore

To decrypt the keystore, you must first verify that the entered passphrase is correct. Based on the entered passphrase, a new Derived Key and MAC are generated and checked for a match with the MAC within the keystore. If a match is confirmed, the new Derived Key, Cipher text, and cipher parameters information within the keystore are input into the AES decryption algorithm to decrypt the ciphertext into the private key, as shown in Fig. 4 [20].

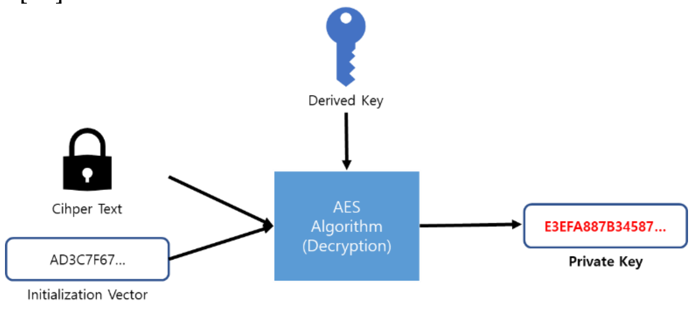


Figure 5. Decrypt Cipher

2.5.2. Symverse

Symverse is a blockchain platform with near 1-second finality based on a self-sovereign distributed ID system. It uses a unique 10-byte ID system with an ID document which contains 20-byte Public Key Hash used as an address and a 10-byte ID system consisting of a network identifier (SymID, 2 bytes), CitizenID (6 bytes), and account identifier (2 bytes). Symverse is a collaborative blockchain service that can be extended to independent blockchain platforms based on the Symverse platform. Its block creation method is an enhanced BFT (Byzantine Fault Tolerant) appling strategic voting theory and PoS (Proof of Stake) to achieve fast block finality [21].

- SCT-20

SCT (Symverse Contract Template) is a template protocol designed to make it easy to create and operate smart contracts within the Symverse blockchain. In Ethereum, the ERC-20 protocol involves writing smart contracts in Solidity and registering them on the blockchain, allowing them to be operated by the EVM. However, in Symverse, SCT-20 provides standard input and output, as shown in Table 1, and allows for the creation of token smart contracts via an RPC protocol in JSON format, making it simple to use. To create and use an SCT contract, SYM coins, which are used as gas, are required, and calling SCT functions consumes SYM coins as shown in Table 2.

Table 1. SCT20 creation parameters

Parameters	Type	Description
Name	Address	Smart Contract (Token) Name
Symbol	String	Smart Contract (Token) Symbol. The length should be from 3 to 10
Amount	Int	Total Supply
Owner	Address	10 Bytes - address of the contract owner

Table 2. SCT20 operation gas

Type	Function	Description	SCT Gas	Total Gas

SCT20	SCT20_CREATE	Create SCT20 contract	49,000	8,049,000
	SCT20_TRANSFER	Transfer token	7,000	56,000
	SCT20_TRANSFER_FROM	Token delegation transfer	9,000	58,000
	SCT20_MINT	Issue additional token	7,000	56,000
	SCT20_BURN	Token burn	7,000	56,000
	SCT20_PAUSE	Contract suspension	4,000	53,000
	SCT20_UNPAUSE	Resume suspended contract	4,000	53,000

- Transactions

A transaction is an act of recording ledger within a block on the blockchain, and once a transaction is recorded, it cannot be modified or deleted. In Symverse, there are three types of transactions: general transaction, SCT transaction, and deposit transaction [24]. The required data in a transaction is as shown in Table. 3.

Table 3. Transaction data parameters

Field	Type	Description
from	address	[10] bytes, sender address
nonce	int	the count of transaction publication in the account
gasPrice	int	gas price per gas unit
gas	int	gas amount for executing transaction
to	address	[10] bytes, receiver address or contract address or nil
value	int	the value sent with this transaction or amount of deposit
input	data	[] byte, rlp encoded data (contract or sct)
type	int	transaction type (0: general(default), 1: sct, 2: deposit)
workNodes	[] address	array, list of work nodes what deliver the transaction (count == 1)
extraData	int	[] byte

The gas consumed in a transaction is calculated by the following formula:

```
gas = base_gas
      + (number of none-zero-byte)*680
      + (number of zero-byte)*40
      + contract_operation_gas
```

Figure 6. Calculate consume gas

Transactions can be processed in the order shown in Fig.7, allowing for the execution of one transaction.

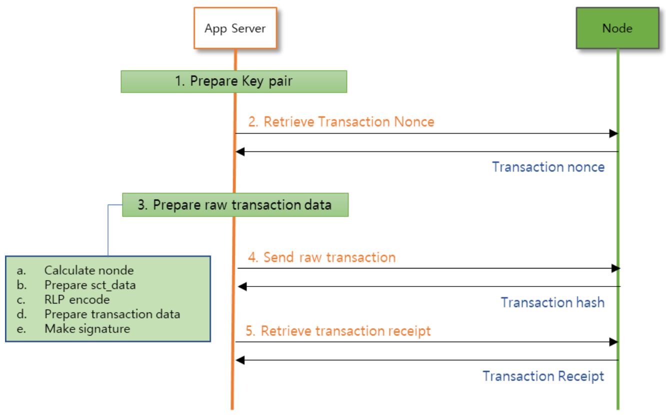


Figure 7. Symverse transaction processing flow [22]

The transaction Raw DATA is sent to nodes through a signature procedure,

```
step 1. Tx = tx_data + {chain_id, "", ""}
step 2. encoded_Tx = RLP_encode(Tx)
step 3. Tx_hash = SHA3(encoded_Tx)
step 4. V, R, S = SIGN(Tx_hash)
step 5. signed_Tx = (tx_data , V, R, S)
step 6. RLP_encode(signed_Tx)
step 7. send message
```

Figure 8. Transaction signing process

The executed transaction returns a Transaction Receipt, which can be used to verify the information from blockchain data in the node.

The next chapter will introduce the details and architecture of the proposed system in this paper, as well as various modules that solve the problems discussed in this chapter. In addition, we will provide an evaluation of the system's performance, security, and usability, demonstrating the potential to provide a comprehensive solution for non-face-to-face financial transactions.

3. Token Payment Blockchain System

The proposed token payment blockchain system is a secure system that enables real-time token transfers between individuals or parties without exposing personal information. Once the transaction is completed, the transaction history can be verified by anyone using a wallet or using a separate blockchain scanner. This architecture consists of five modules: a module for wallet owners to generate and store private keys in the keystore, a module for checking and charging insufficient gas in the wallet, a module for sending and receiving tokens, a module for DDoS defense of the system, and a module for real-name authentication of the keystore.

3.1. Overall Architecture

The blockchain system consists of key management module, token management module, token payment module, DDos defense module, and real-name authentication module for legacy system. The key management module efficiently generates, changes, and retrieves private keys in conjunction with the keystore, while the token management module manages tokens by creating, querying, minting, and burning them. The token payment module automatically charges and sends gas when a general user has no gas. The DDos defense module is composed of distributed attack prevention and blocking of overseas access to protect the server, and the real-name authentication module authenticates the wallet according to domestic law. Figure 9 shows the module-by-module configuration and interconnection of the system that enables P2P real-time payments based on the server.

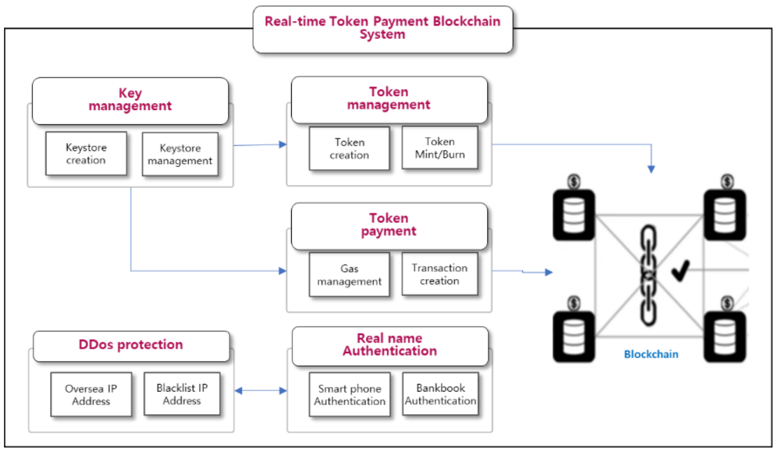


Figure 9. Token payment blockchain platform

3.2. *The control flow of the proposed system*

The control flow of the proposed system is as follows: When the user requests to send tokens by entering the blockchain address and token amount of the recipient on their smartphone, the key management module reads and decrypts the user's keystore to extract their private key for transaction signing. The signed transaction is then sent to the token transfer module. When the token payment is made, the gas of the sender's blockchain address, SymID, is checked. If the gas is insufficient, the gas is automatically recharged to sender's SymID account and the transaction is sent to the nodes of the blockchain. The transaction sent to the blockchain in the nodeis recorded in a block and, after going through the PoS (Proof of Stake) consensus, the block is confirmed and permanently stored on the blockchain. When the recipient requests to check their token balance in their wallet, they query the node through the token management module and he receive the balance information data.

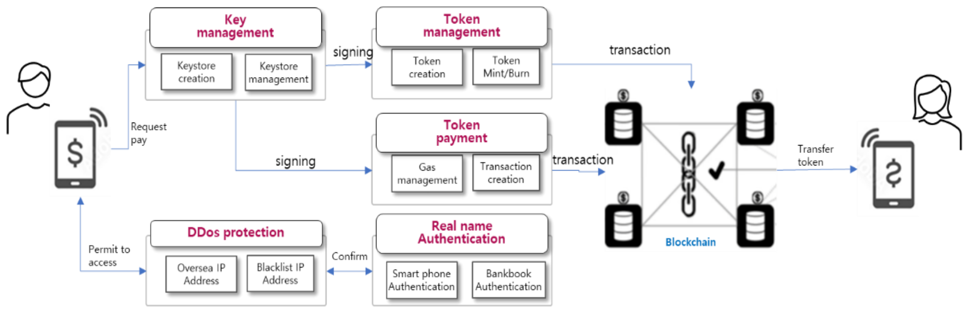


Figure 10. Control Flow of System

3.3. *Key management module*

The key management module generates a Private Key as a 32-byte hexa code by creating a 256-bit random number when the user requests to create a wallet by entering a passphrase. It also generates a corresponding Public Key and Public Key Hash and stores them in the file system as a keystore. The generated Public Key Hash and the user's ID are sent to the Certificate Authority Server of the Symverse blockchain to create a SymID, which is an address used in the Symverse blockchain, and store it in the DB. Users can send tokens with just their ID and passphrase at login without having to remember their SymID. The passphrase for the keystore can be changed, but the Private Key and SymID cannot be changed. When a user requests to sign a transaction Raw Code, they enter the passphrase to decrypt and extract the Private Key from the keystore, sign the Raw Code, and create a transaction.

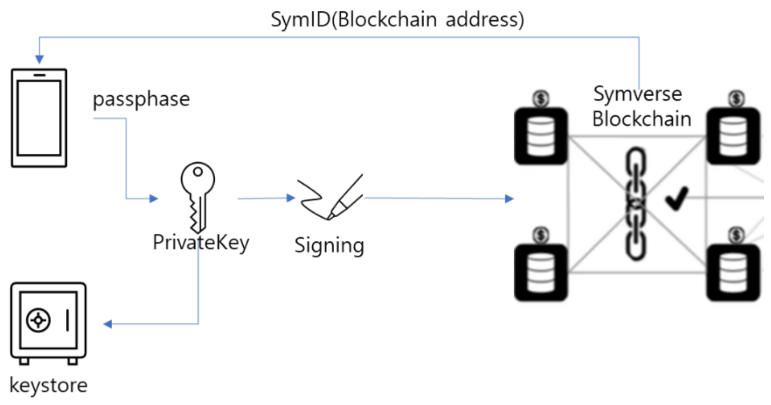


Figure 11. Transaction creation process through private key signature

3.4. Token management module

The token management module is responsible for creating tokens that represent the initial supply quantity and can mint or burn the token amount as needed. Token minting on the Symverse blockchain can be done using the SCT-20 template, which has a minting function to increase the total token issuance amount and a Burn function to decrease it. To create a token, you need to input the token symbol name, symbol, total issuance quantity, owner SymID according to the JSON convention of SCT-20 and sign the generated bytecode with the private key extracted from the keystore using a passphrase. The bytecode is then converted into a transaction and permanently registered on the blockchain. To retrieve information such as the token contract address, you can query the blockchain using the hash value of the returned transaction receipt.

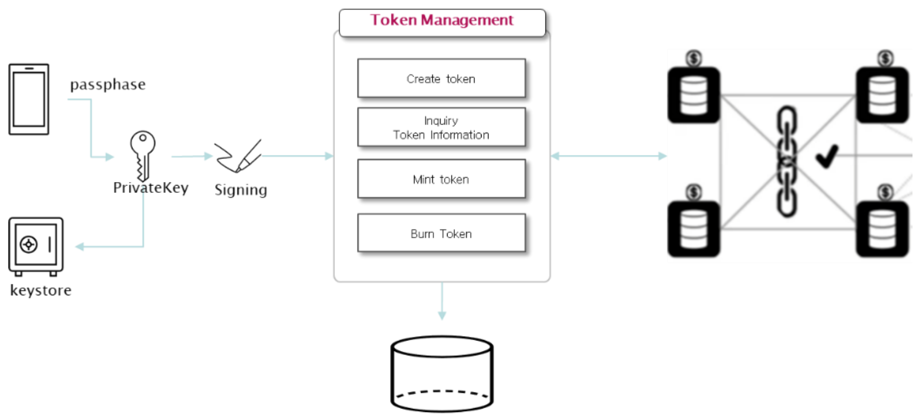


Figure 12. Token management module

3.5. Token transfer module

The Token Transfer Module is responsible for transferring tokens between users. When a user sends tokens, gas is consumed to execute the smart contract, and the user must manage the gas separately, making it difficult to use. To make it easier for users, a module is provided so that it manages the gas consumption payment instead of users. To transfer tokens, the user generates bytecode and signs it using their private key. The gas consumed by the transaction is then calculated. If there is no enough gas, the Contract Owner requests that SYM coins has to be automatically recharged to the SymID account on the blockchain. In the next step, the transaction is sent to the node and permanently recorded in the block through consensus algorithm so that the tokens ae transferred to the recipient.

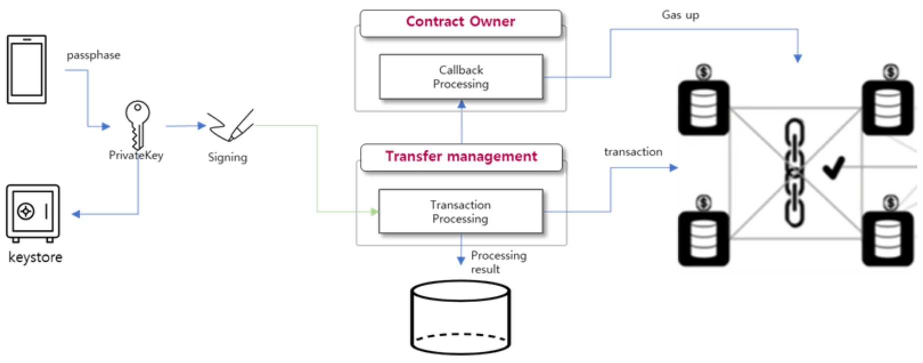


Figure 13. Token transfer module

3.6. DDos protection module

The current token payment system has been developed for the purpose of actual use in Korea, so the module aims to block overseas IPs so as to comply to Korean laws for real-name authentication and to indiscriminate attacks from overseas. Based on the built-in database of global IP address allocation, it extracts IP address allocated to Korea and allows access only if it is included within the Korean IP address range. It also manages DDoS source IPs separately in the database to block access, and effectively defends against DDoS by handling traffic from the built-in database without traffic increase on the main DB during multiple loads.

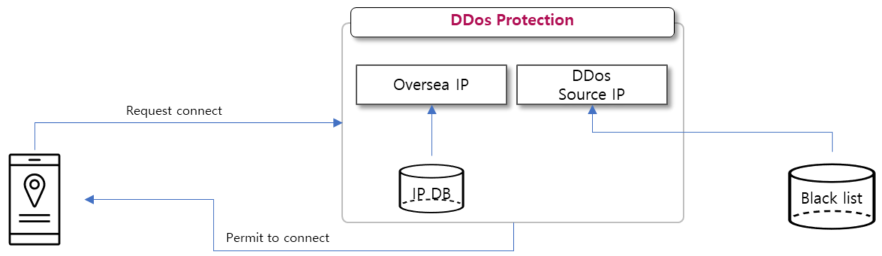


Figure 14. DDos Block Module

3.7. Real name verification module

The real name verification module interfaces with external legacy systems. When a smartphone user enters their name, phone number, date of birth (6 digits), and the first digit to denote the sex code in the resident registration number, the server converts it into the format requested by the related institution and sends a response code to the smartphone number via SMS (Short Messaging System). The user inputs the response code, and the module verifies their real name by receiving a response from the credit information management institution, and performs real-name verification for the keystore.

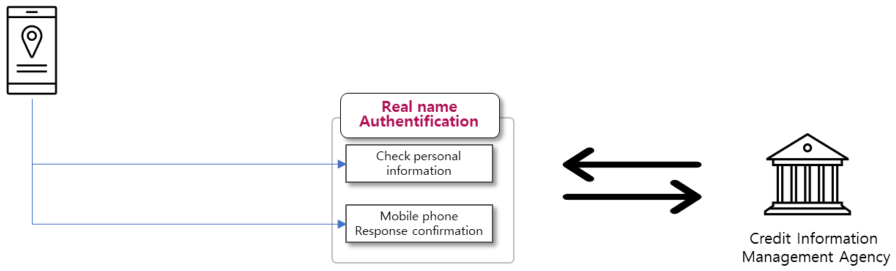

Real name Authentification
Check personal information
Mobile phone Response confirmation
Credit Information Management Agency

Figure 15. real name authentication module

4. Implementation

The proposed system is composed of the commonly used server module, CentOS, and Apache, PHP, and mySQL. The overseas IP blocking database is separately configured using SQLite. The blockchain is integrated with the Symverse mainnet through a service node installed for the users' convenience. Especially, the user interface is programmed by using Flutter language framework provided by Google so that it enables various devices such as smartphones, tablets, and PCs to utilize the proposed system.

4.1. System Environment

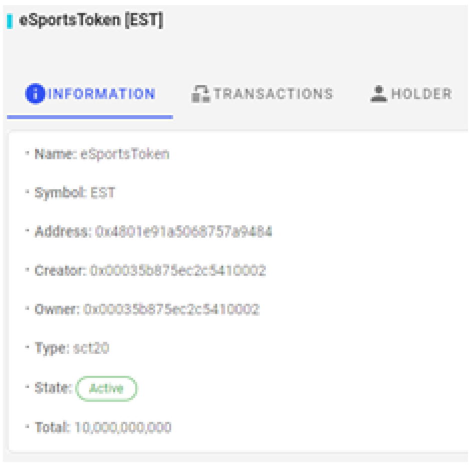
The implementation system consists of a web service server and a blockchain work node, called gsym. The web service is installed with CentOS as the operating system, Maria DB as the database, Apache as the web server, and PHP as the scripting language to implement the REST server. The user interface is developed in Flutter language using Android Studio and is released as a web version uploaded to the server for users' convenience through web browsers. The overseas IP blocking database is implemented using SQLite3 to avoid affecting the internal logic of the main database. The table below lists the environments and libraries used in the implementation.

Table 4. List of environments and libraries

OS	Centos 7.4.17	Database	mySQL 5.7.36 sqlite3
Language	PHP/HTML/Dart/Javascript	Webserver	Apache 2.4.52
Webserver-plugin	Php 7.4.1 Scrypt 1.4.2	Blockchain	symverse
UI framework	Flutter/Bootstrap		gsym
Test tool	JMeter V5.5	Browser	Chrome 버전 111.0.5563.66(공식 빌드) (64비트)

4.2. Token

The EST token used in the system implementation obtained a smart contract ID of "0x4801e91a5068757a9484" generated through SCT-20, and can be reviewed through the Symverse Scanner as shown in Fig.16. (https://scan.symverse.org/v2/sct2x_token/SCT20/0x4801e91a5068757a9484)


eSportsToken [EST]

- Name: eSportsToken
- Symbol: EST
- Address: 0x4801e91a5068757a9484
- Creator: 0x00035b875ec2c5410002
- Owner: 0x00035b875ec2c5410002
- Type: sct20
- State: Active
- Total: 10,000,000,000

Figure 16. EST Token Information

The token minting process utilizes on Symverse's SCT. The minting process is initiated by creating information about the token in an array called sct_data, as shown in Fig.17.

```
sct_data = [
   "0x14", // SCT20
"0x0", // SCT20_CREATE Command
[
       "eSportsToken", // Token Name
       "EST", // Token Symbol
       "0x204fce5e3e25026110000000",    // Token Amount    (1 Billion)
       "0x00035b875ec2c5410002" // Owner's SymID
   ]
]
```

Figure 17. Array of SCT-20 input parameters.

Sct_data is processed by RLP (Recursive Length Prefix) Encoding and the result is as shown in Fig.18.

```
0xec1480e98c6553706f727473546f6b656e834553548c204fce5e3e250261100000008a00035b875ec2c54100
```

Figure 18. Sct_data RLP Encoding Result

The transaction is signed and sent to the work node and the result can be verified after block is confirmed, as shown in the example of Fig.19.

```
(
      [jsonrpc] => 2.0
      [id] => 1
      [result] => Array
            (
                  [creator] => 0x00035b875ec2c5410002
                  [name] => eSportsToken
                  [owner] => 0x00035b875ec2c5410002
                  [state] => active
                  [stateCode] => 0x0
                  [symbol] => EST
                  [total] => 0x204fce5e3e25026110000000
                  [type] => sct20
            )
)
```

Figure 19. SCT-20 Information

4.3. Database

The database consists of tables for user information, wallet, and transfer, as well as tables for token, real-name authentication logs, and IP for DDoS defense. Table 5 summarizes the database table list for system implementation to operate a service.

Table 5. List of Database Tables

Table Name	Table ID
User	nr_user
Wallet	est_wallet
Transfer	nr_charge
Charge	nr_deposit
Withdraw	nr_withdraw
Statics Transfer	nr_settle_sale
Statics Join	nr_stat_join
Realname log	nr_log_sms
DDos ban ip	nr_ddos_ip

4.4. User interface of client side

The user interface of the client used by wallet users is organized as shown in Table 6.

Table 6. User's menu

Menu		Description
class1	class2	
Client	Login	Login with member ID and password
	HP authentication	Mobile phone real name verification
	Charge	Token charging function for payment
	Charge list	Recently charged history list
	Transfer	Transfer Listings for sending to sellers
	Transfer list	List of sent history records

The interface for the wallet user who receives payment is composed of menus that allow them to view transaction history (as in Table.7) and exchange the received tokens into fiat currency.

Table 7. Seller's Menu

Menu		Description
class1	class2	
Seller	Transfer list	Deposits and withdrawals of tokens and error history

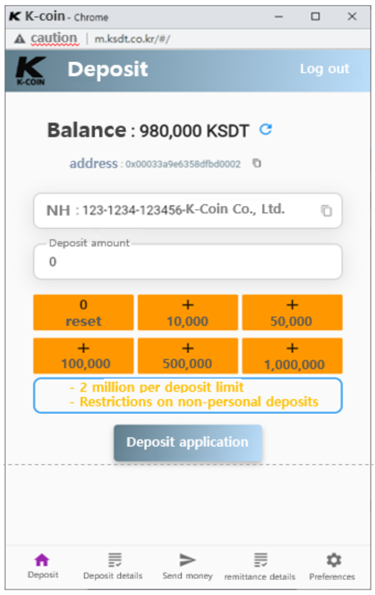
	Convert list	Details of application for conversion and details of results
	Convert application	Application for changing safe traded tokens to fiat

The platform operator's interface is configured with menus such as statistics management, member management, and deposit/withdrawal management, as shown in Table.8.

Table 8. Platform operator's menu

Menu		Description
Class1	Class2	
Platform	statistics	Member registration status by time/day/month
	Member management	Member list/real name authentication management
	Settlement management	Settlement information/daily settlement/settlement statistics
	Deposit and Withdrawal Management	Conversion list/Charging list/Real-time notification
	Mobile gift certificate management	Mobile gift certificate sales list/purchase list
	manager management	Admin list and administrator privilege setting

To convert fiat currency to tokens in order to transfer tokens, users make a request for conversion through an interface dipicted in Fig.20, which shows the actual usage example.



Balance : 980,000 KSDT

address : 0x00033a9e6358dfbd0002

NH : 123-1234-123456-K-Coin Co., Ltd.

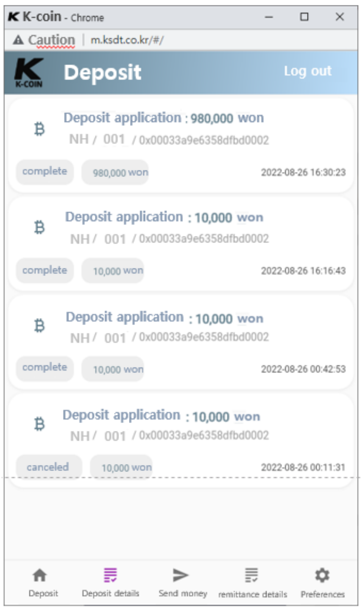
Deposit amount: 0

- 2 million per deposit limit
- Restrictions on non-personal deposits

Figure 20. Interface of Convert Token

doi:10.20944/preprints202305.0740.v1

The history of conversion transaction can be confirmed in the deposit history as shown in Fig.21, and it checked whether the conversion has been completed.


K-coin - Chrome
Caution | m.ksdt.co.kr/#/
Deposit Log out

Deposit application : 980,000 won
NH / 001 / 0x00033a9e6358dfbd0002
complete 980,000 won 2022-08-26 16:30:23

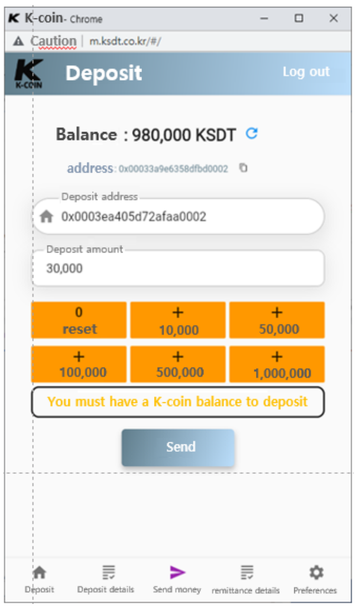
Deposit application : 10,000 won
NH / 001 / 0x00033a9e6358dfbd0002
complete 10,000 won 2022-08-26 16:16:43

Deposit application : 10,000 won
NH / 001 / 0x00033a9e6358dfbd0002
complete 10,000 won 2022-08-26 00:42:53

Deposit application : 10,000 won
NH / 001 / 0x00033a9e6358dfbd0002
canceled 10,000 won 2022-08-26 00:11:31

Figure 21. History of Convert Status

After depositing tokens, users can transfer tokens to a specified seller after confirming the transfer address, using the Fig.22 interfaces.

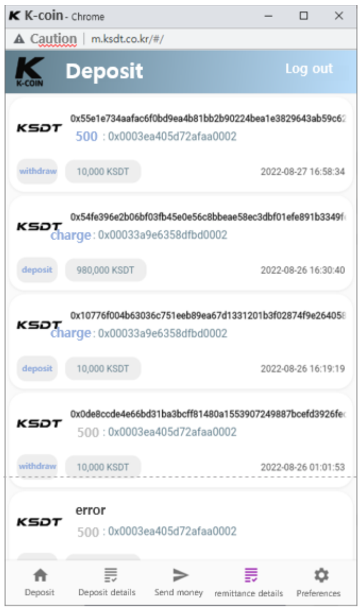

K-coin- Chrome
Caution | m.ksdt.co.kr/#/
Deposit Log out

Balance : 980,000 KSDT
address: 0x00033a9e6358dfbd0002
Deposit address: 0x0003ea405d72afaa0002
Deposit amount: 30,000

You must have a K-coin balance to deposit

Figure 22. Interface of Transfer Token

All the transaction history (deposit/convert/payment) are displayed as a list as shown in Fig.23 to represent the result of the transaction.


K-coin - Chrome
Caution | m.ksdt.co.kr/#/
Deposit
Log out

KSDT
0x55e1e734aafac6f0bd9ea4b81bb2b90224bea1e3829643ab59c62
500 : 0x0003ea405d72afaa0002
withdraw 10,000 KSDT 2022-08-27 16:58:34

KSDT
0x54fe396e2b06bf03fb45e0e56c8bbeae58ec3dbf01efe891b3349f
charge : 0x00033a9e6358dfbd0002
deposit 980,000 KSDT 2022-08-26 16:30:40

KSDT
0x10776f004b63036c751eeb89ea67d1331201b3f02874f9e264058
charge : 0x00033a9e6358dfbd0002
deposit 10,000 KSDT 2022-08-26 16:19:19

KSDT
0x0de8ccde4e66bd31ba3bcff81480a1553907249887bcefd3926fe
500 : 0x0003ea405d72afaa0002
withdraw 10,000 KSDT 2022-08-26 01:01:53

KSDT
error
500 : 0x0003ea405d72afaa0002

Figure 23. History of Transfer token

4.5. External interface

External interfaces are represented by three interfaces as shown in Table.9, and each interface communicates using the JSON RPC method.

Table 9. External Interface

I/FID	Sender	Receiver	Interface Description	Data format
IF _001	KSDT server	gsym	Symverse work node	JSON
IF _002	Credit Rating Agency	Ok-names	Ok-names real name inquiry	JSON
IF_003	Symverse	Symverse CA	Issue SymID	JSON

5. Experiment and Results

To evaluate the proposed system, we measured it using JMeter, which uses the BMT tool to test the HTTP protocol.

5.1. System environment

For performance evaluation, the system environment consists of a physical node with a main web server environment with one CPU (AMD Ryzen 5 PRO 3400GE) and 8GB DDR4 main memory and an SSD of 240GB, a physical node as a Symverse work node with server environment consisting of one CPU (Intel(R) Xeon(R) CPU E3-1230 v6 @ 3.50GHz) and 8GB DDR4 main memory and an SSD of 240GB, and a client system with one CPU (Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz) and DDR4 16GB memory so that it can perform the evaluation of various experiments.

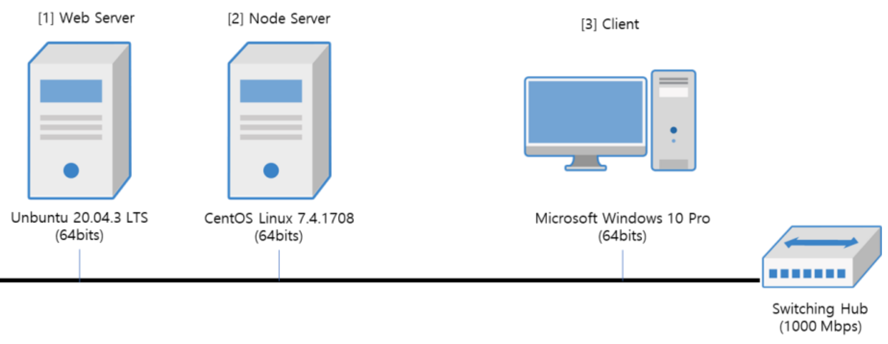

[1] Web Server
[2] Node Server
[3] Client
Unbuntu 20.04.3 LTS (64bits)
CentOS Linux 7.4.1708 (64bits)
Microsoft Windows 10 Pro (64bits)
Switching Hub (1000 Mbps)

Figure 24. Experiment Environment

Table 10. Test environment

No	Role	OS	CPU	Memory	Disk	spec
1	Web Server	Ubuntu 20.04.3 LTS	AMD Ryzen 5 PRO 3400GE	8G	SSD 256GB	-mySQL 5.7.36
						-Apache 2.4.52
						-PHP 7.4.1
						-sqlite3
2	Node Server	CentOS Linux release 7.4.1708	Intel(R) Xeon(R) CPU E3-1230 v6 @ 3.50GHz	8G	SSD 512GB	-gSym
3	Client	Windows 10 Pro(64bit)	Intel core i7-8750H 2.20Ghz	16G	SSD 512GB	-chrome V111.0.5563.65
						-jMeter v5.5

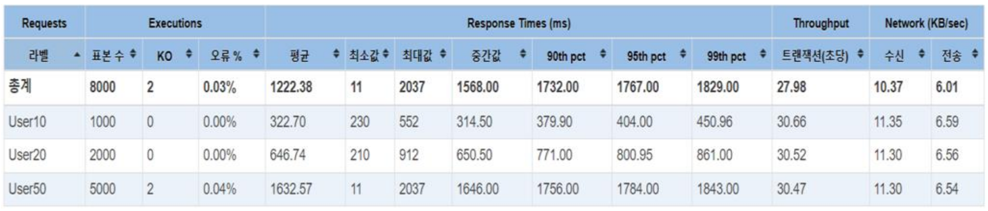
5.2. Performance testing

The token payment system is a system that provides services through JSON format using RPC, not web page calls, as it is composed of a web application server. We implement performance testing on wallet creation response and balance inquiry. where both systems call and result are representing as a simple JSON format.

5.2.1. Wallet creation response time

Wallet creation requests are sent to the server and the results are stored in the database. First, the results up to key store creation are tested. Then, to measure the pure wallet creation time, the IP blocking for DDoS prevention is turned off. For the user10 scenario, assuming a loop of 100 requests with 10 threads, the ramp-up is set to 0, allowing only up to 10 requests per second. For the user20 scenario, assuming a loop of 100 requests with 20 threads, the ramp-up is set to 0, allowing only up to 20 requests per second. For the user50 scenario, assuming a loop of 100 requests with 50 threads, the ramp-up is set to 0, allowing only up to 50 requests per second. The summary of the test results for the three scenarios is shown in Table 11.

Table 11. TC1 Response time of keystore creation (without DDos protection)



Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
라벨	표본 수	KO	오류 %	평균	최소값	최대값	중간값	90th pct	95th pct	99th pct	트랜잭션(초당)	수신	전송
총계	**8000**	**2**	**0.03%**	**1222.38**	**11**	**2037**	**1568.00**	**1732.00**	**1767.00**	**1829.00**	**27.98**	**10.37**	**6.01**
User10	1000	0	0.00%	322.70	230	552	314.50	379.90	404.00	450.96	30.66	11.35	6.59
User20	2000	0	0.00%	646.74	210	912	650.50	771.00	800.95	861.00	30.52	11.30	6.56
User50	5000	2	0.04%	1632.57	11	2037	1646.00	1756.00	1784.00	1843.00	30.47	11.30	6.54

Average response time corresponding to the variation of the number of threads are depicted in Fig.25.

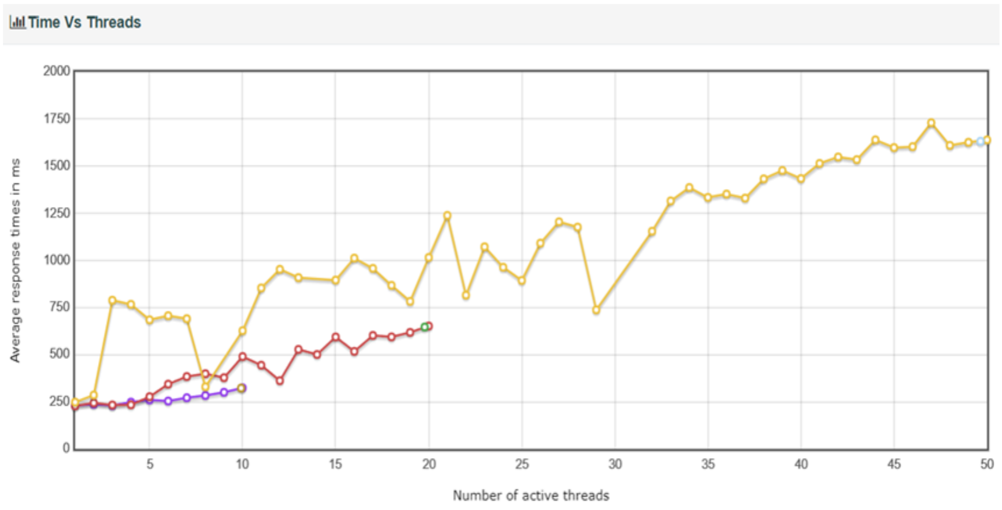
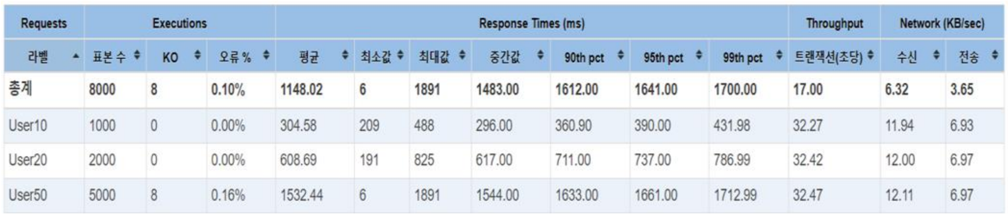


Figure 25. TC1 Active Number of Thread vs Response Time (without DDos protection)

Under the same conditions, the results of processing with DDos blocking turned off are shown in Table 12. The DDos blocking logic showed an impact of approximately 2 transactions per second, with 1.61 for User10, 1.9 for User20, and 2.0 for User50, in terms of transactions per second. However, for User50, there was a significant variation in response time.

Table 12. TC2 Response time keystore creation (with DDos protection)



Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
라벨	표본 수	KO	오류 %	평균	최소값	최대값	중간값	90th pct	95th pct	99th pct	트랜잭션(초당)	수신	전송
총계	**8000**	**8**	**0.10%**	**1148.02**	**6**	**1891**	**1483.00**	**1612.00**	**1641.00**	**1700.00**	**17.00**	**6.32**	**3.65**
User10	1000	0	0.00%	304.58	209	488	296.00	360.90	390.00	431.98	32.27	11.94	6.93
User20	2000	0	0.00%	608.69	191	825	617.00	711.00	737.00	786.99	32.42	12.00	6.97
User50	5000	8	0.16%	1532.44	6	1891	1544.00	1633.00	1661.00	1712.99	32.47	12.11	6.97

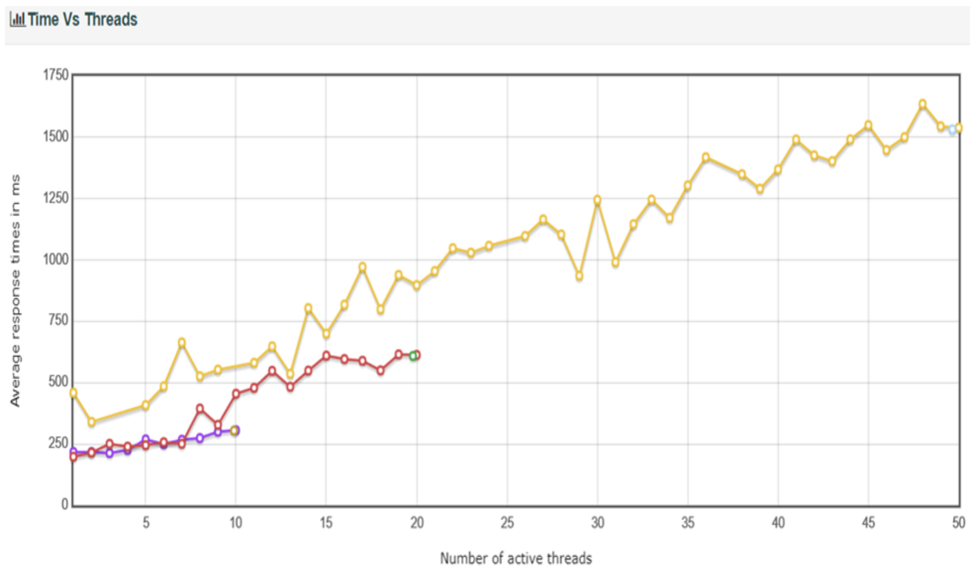
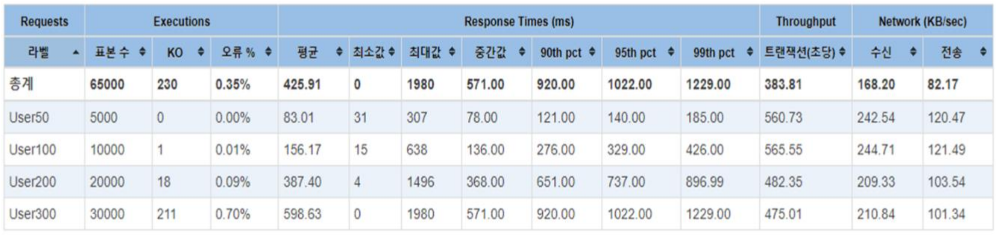


Figure 26. TC2 Active Number of Thread vs Response Time (with DDos protection)

5.2.2. Balance inquiry response time

W Balance response is the process of querying SymID from the blockchain node and delivering the balance to the client in JSON format. To measure the balance response time, DDos defense IP blocking was turned off, and a scenario was assumed where user50 generates 50 threads with 100 loops of requests, setting the ramp-up to 0 so that only up to 50 requests can be made per second. User100 assumes a scenario where 100 threads with 100 loops of requests are generated, setting the ramp-up to 0 so that only up to 100 requests can be made per second. User200 assumes a scenario where 200 threads with 100 loops of requests are generated, setting the ramp-up to 0 so that only up to 200 requests can be made per second. User300 assumes a scenario where 300 threads with 100 loops of requests are generated, setting the ramp-up to 0 so that only up to 300 requests can be made per second. The summary report of the test is shown in Table.13. The transactions per second (TPS) were 560.73 without errors in 50 threads, 565.55 TPS with 0.01% errors in 100 threads, 482.35 TPS with 0.09% errors in 200 threads, and 475.0 TPS with 0.7% errors in 300 threads, indicating an increase in errors as the number of threads increases.

Table 13. TC3 Response time of balance inquiry (without DDos protection)



Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
라벨	표본 수	KO	오류 %	평균	최소값	최대값	중간값	90th pct	95th pct	99th pct	트랜잭션(초당)	수신	전송
총계	**65000**	**230**	**0.35%**	**425.91**	**0**	**1980**	**571.00**	**920.00**	**1022.00**	**1229.00**	**383.81**	**168.20**	**82.17**
User50	5000	0	0.00%	83.01	31	307	78.00	121.00	140.00	185.00	560.73	242.54	120.47
User100	10000	1	0.01%	156.17	15	638	136.00	276.00	329.00	426.00	565.55	244.71	121.49
User200	20000	18	0.09%	387.40	4	1496	368.00	651.00	737.00	896.99	482.35	209.33	103.54
User300	30000	211	0.70%	598.63	0	1980	571.00	920.00	1022.00	1229.00	475.01	210.84	101.34

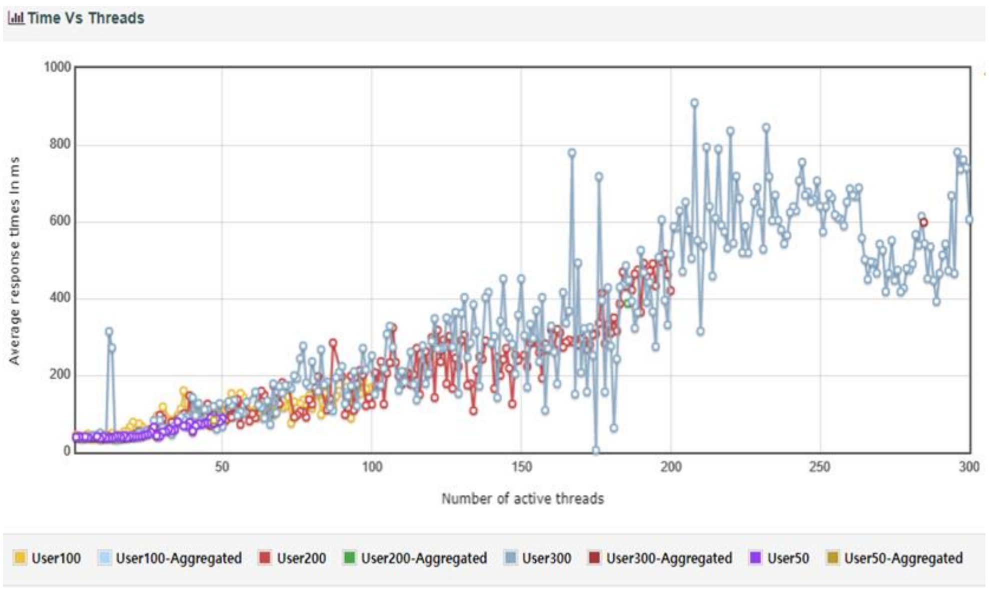
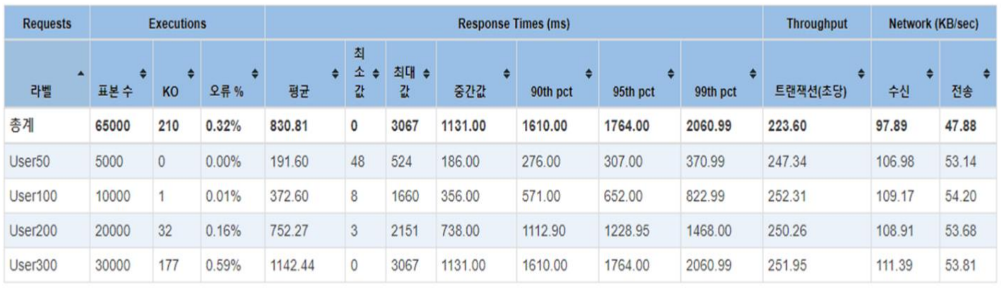


Figure 27. TC3 Active Number of Thread vs Response Time (without DDos protection)

Under the same conditions with DDos blocking turned on, the results are as follows. The DDos defense logic has an impact of approximately 220 TPS in the number of transactions per second. Specifically, in User50, the TPS decreased from 560.73 to 247.34. In User100, it decreased from 565.55 to 252.31. In User200, it decreased from 482.36 to 250.26. In User300, it decreased from 475.01 to 251.95.

Table 14. TC4 Response time of balance inquiry (with DDos protection)



Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
라벨	표본 수	KO	오류 %	평균	최소값	최대값	중간값	90th pct	95th pct	99th pct	트랜잭션(초당)	수신	전송
총계	**65000**	**210**	**0.32%**	**830.81**	**0**	**3067**	**1131.00**	**1610.00**	**1764.00**	**2060.99**	**223.60**	**97.89**	**47.88**
User50	5000	0	0.00%	191.60	48	524	186.00	276.00	307.00	370.99	247.34	106.98	53.14
User100	10000	1	0.01%	372.60	8	1660	356.00	571.00	652.00	822.99	252.31	109.17	54.20
User200	20000	32	0.16%	752.27	3	2151	738.00	1112.90	1228.95	1468.00	250.26	108.91	53.68
User300	30000	177	0.59%	1142.44	0	3067	1131.00	1610.00	1764.00	2060.99	251.95	111.39	53.81

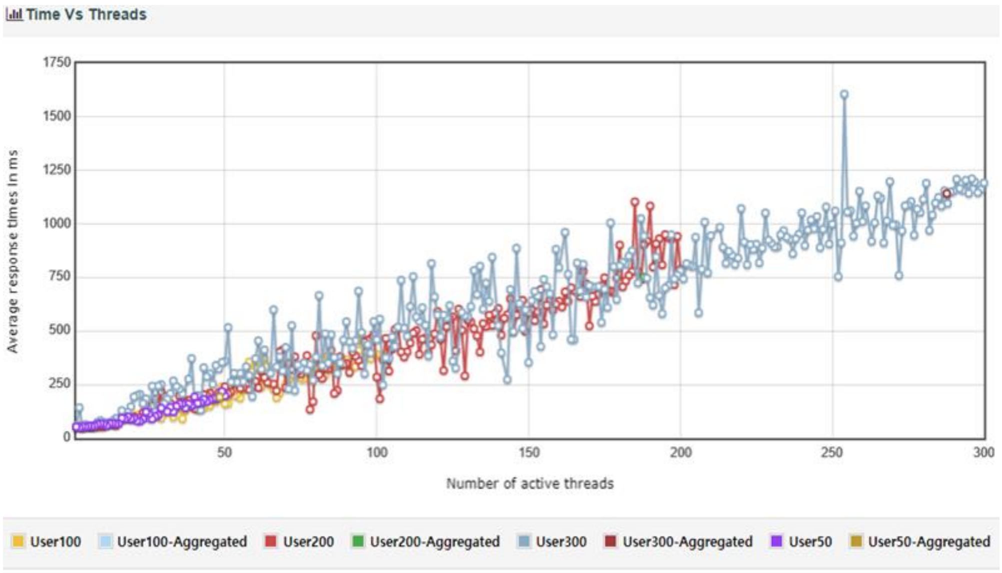


Figure 28. TC4 Active Number of Thread vs Response Time (with DDos protection)

6. Conclusion

This paper has proposed a new server-based real-time P2P token payment blockchain system designed to address the problems of non-face-to-face transactions in the financial sector, especially in the situation where sophisticated voice phishing is spreading and reliance on open banking systems is increasing. This solution overcomes the limitations of private key management difficulties and gas fee management while utilizing the advantages of blockchain technology.

The server-based key management module effectively minimizes the risk of loss by securely storing private keys in keystore files and databases. The server-based token management module simplifies token generation and management, ensuring smooth token transactions through automatic gas charging. The transaction verification module, which utilizes transaction IDs without exposing personal information, guarantees transaction integrity and non-repudiation. In addition, advanced security measures such as external access IP blocking and DDoS protection are introduced to keep user data safe.

It also aims to provide convenient, secure and accessible online payment solutions to the general public by implementing identity verification using smartphones and platform-agnostic web applications. This comprehensive approach has the potential to innovate non-face-to-face transactions in the financial sector by providing user-friendly and secure alternatives to traditional methods. As the digital environment continues to evolve, the non-contact P2P real-time tokenized payment system can become an important tool for protecting users from financial fraud and creating a more inclusive and flexible financial ecosystem.

Through this research, it was demonstrated that blockchain token payments can be utilized as an anti-voice phishing technology by blocking the possibility of exposure to crime through transparent transactions where all details can be made public and both the payer and the payee can be verified. Third-party authentication for non-repudiation is also possible. Furthermore, it is expected to create business opportunities for innovative finance, such as IoT automatic regular payments, small loan services, and 24-hour uninterrupted payment services. The feasibility and efficiency of the proposed system were confirmed through experimental results, and it is expected to have a significant impact on the financial industry.

References

1. Jain, Ankit Kumar, and B. B. Gupta. "A survey of phishing attack techniques, defence mechanisms and open research challenges." Enterprise Information Systems 16.4 (2022): 527-565.
2. Jeong, Young Ho, Ha, Hyung Joon 2. “Messenger Phishing Crime: Trends and Responses.” Criminal Investigation Studies 8.1 (2022): 31-54.
3. Tama, Bayu Adhi, et al. "A critical review of blockchain and its current applications." 2017 International Conference on Electrical Engineering and Computer Science (ICECOS). IEEE, 2017.
4. Gad, Ahmed G., et al. "Emerging trends in blockchain technology and applications: A review and outlook." Journal of King Saud University-Computer and Information Sciences (2022).
5. Shah, Kaushal, et al. "Exploring applications of blockchain technology for Industry 4.0." Materials Today: Proceedings 62 (2022): 7238-7242.
6. Ahmed, Mohammad Rasheed, et al. "Blockchain based architecture and solution for secure digital payment system." ICC 2021-IEEE International Conference on Communications. IEEE, 2021.
7. Villanueva, Neil Efren. "Blockchain Technology Application: Challenges, Limitations and Issues." Journal of Computational Innovations and Engineering Applications Vol 5.2 (2021).
8. Cho, S. I. "A Study on Factors Affecting the Intention to Use of New Access Media in e-Financial Transactions." Graduate School of Soongsil University, Doctoral Dissertation (2018).
9. Seibert, Jeffrey, and Michael Ducker. "Inter-application delegated authentication." U.S. Patent No. 11,025,624. 1 Jun. 2021.

10. Hinkes, Andrew M. "Throw away the key, or the key holder? coercive contempt for lost or forgotten cryptocurrency private keys, or obstinate holders." Nw. J. Tech. & Intell. Prop. 16 (2018): 225.
11. Yang, Rui. "Development and Supervision of Financial Technology Based on Blockchain." Computational Intelligence and Neuroscience 2022 (2022).
12. Chen, Yan, and Cristiano Bellavitis. "Decentralized finance: Blockchain technology and the quest for an open financial system." Stevens Institute of Technology School of Business Research Paper (2019).
13. Ostern, Nadine Kathrin, and Johannes Riedel. "Know-your-customer (KYC) requirements for initial coin offerings." Business & Information Systems Engineering 63.5 (2021): 551-567.
14. Sable, Nilesh P., et al. "The Secure E-Wallet Powered by Blockchain and Distributed Ledger Technology." 2022 IEEE Pune Section International Conference (PuneCon). IEEE, 2022.
15. Praitheeshan, Purathani, et al. "Attainable hacks on Keystore files in Ethereum wallets—a systematic analysis." Future Network Systems and Security: 5th International Conference, FNSS 2019, Melbourne, VIC, Australia, November 27–29, 2019, Proceedings 5. Springer International Publishing, 2019.
16. Urien, Pascal. "Innovative Wallet Using Trusted On-Line Keystore." 2021 3rd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS). IEEE, 2021.
17. Lehto, Niko, et al. "CryptoVault-A Secure Hardware Wallet for Decentralized Key Management." 2021 IEEE International Conference on Omni-Layer Intelligent Systems (COINS). IEEE, 2021.
18. Phuc, Tran Song Dat, and Changhoon Lee. "Password Hashing Algorithms-From Past to Future." JOURNAL OF PLATFORM TECHNOLOGY 3.4 (2015): 63-75.
19. Antonopoulos, Andreas M., and Gavin Wood. Mastering ethereum: building smart contracts and dapps. O'reilly Media, 2018.
20. Symverse. (2022). SymVerse Whitepaper. https://symverse.com/en/learn/whitepaper/.
21. Symverse SCT Guide. https://github.com/symverse-lab/Document/wiki/SCT-Guide