Implementation of the Cross-Blockchain Interacting Protocol

Rita Tsepeleva and Vladimir Korkhov(✉)

Saint Petersburg State University, St. Petersburg, Russia
st062153@student.spbu.ru, v.korkhov@spbu.ru

Abstract. Blockchain is a developing and promising technology that can provide users with such advantages as decentralization, data security and transparency of transactions. Blockchain has many applications, one of them is the decentralized finance (DeFi) industry which is growing more and more recently. The concept of decentralized finance involves the creation of a single ecosystem of many blockchains that interact with each other. The problem of combining and interacting blockchains becomes crucial to enable DeFi. In this paper, we look at the essence of the DeFi industry, the possibilities of overcoming the problem of cross-blockchain interaction, present our approach, and analyze the results of the proposed solution.

Keywords: Blockchain · Distributed ledger technologies · Solidity · Smart-contracts · Decentralized finances

1 Introduction

Blockchain is a modern technology that is a huge distributed database, i.e. a database whose components are placed in various nodes of a computer network in accordance with certain criteria. They reflect the transfer of information from one user to another. This database is stored on a large number of computers and has a decentralized character. It means that there is no central node that manages everything. It is a great advantage, because if there is a failure of one machine in the system, the entire system will continue to work properly and the information will not be lost. The data is organized into chain of blocks, each new block contains encrypted information from the previous block. Such an organization ensures that no data can be replaced, corrected or deleted. Accordingly, the information is as reliable and secure as possible.

Thanks to these advantages, blockchain is popular in the field of finance and in other spheres. In addition to the obvious advantages of blockchain in the form of reliability, transparency and etc., blockchain technology provides another very important thing. This is the ability to write smart contracts, i.e. executable code that helps users manage their finances independently, without resorting to the help of third parties (for example, the banking sector). However, there is an indisputable fact that smart contracts are not completely safe to use, because a

O. Gervasi et al. (Eds.): ICCSA 2021, LNCS 12952, pp. 42–55, 2021.
https://doi.org/10.1007/978-3-030-86973-1_4

mistake in the code can cost a lot of money. Despite this fact, the industry of decentralized finance is becoming more and more popular.

Next, we will take a closer look at decentralized finances concept, will talk about the most important problem in this area, which in a certain way restricts the use of its tools. We will discuss possible solutions of the problem and will analyze them, as well as give our own version of the solution and compare the results.

The rest of this paper is organized as follows. Section 2 describes the essence of decentralized finance, as well as the problems that need to be solved. Section 3 describes the existing methods and solutions. Section 4 describes the idea of our solution and its implementation. And in Sect. 5, the resulting solution is analyzed in comparison with the existing ones. Section 6 concludes the paper and points out directions of future work.

2 Decentralized Finances and Its Problems

Decentralized Finances (DeFi [1]) is an independent financial ecosystem that gives users a full control over their money without the involvement of governments and banks. Blockchain technologies have played a key role in the implementation of this industry.

2.1 The Essence of DeFi

Interaction in decentralized ecosystems occurs without intermediaries according to the P2P scheme, that is, market participants independently cope with making transactions. It:

1. Saves your time;
2. Saves your money;
3. Allows you to keep your privacy.

What is the reason for the active popularization of DeFi and why are projects increasingly integrating them into their systems? The monetary monopoly of the state is a problem that is quite difficult to solve. The control that we have over our own savings is only relative, and in recent years this has been felt especially acutely. In many European countries, the population is gradually abandoning cash and switching to electronic payments. Banks, constant monitoring, and lack of anonymity are nuances that are absent in DeFi. App developers cannot influence the money of the participants, and the latter manage their own budget. The main advantages of decentralized finance are:

1. Simplicity and accessibility of technology for ordinary people;
2. Scalability and distribution of the registry; as a result, system is more secure and resistant to technical errors;
3. Lower system maintenance costs and lower fees (or no fees);
4. Versability - DeFi can be used in almost any area of our life.

2.2 Usability of DeFi

Developers have already offered quite a lot of options for using decentralized financing. First of all, of course, in the field of banking services.

Mortgages and Insurance. The absence of intermediaries and transparency are the optimal conditions for issuing mortgage loans and insurance, including medical insurance. This category could include social benefits – such as pensions.

Lending. Classical lending is primarily a mass of restrictions. Salary requirements, sureties, and the many certifications that need to be obtained make these services inaccessible to a whole category of people. A person spent his time for making all of documents well, but still waiting for the result, which can be negative, too. DeFi should solve this problem. Moreover, without intermediaries, lending should be cheaper and more reliable. Instant transactions, transparency and security – what is so lacking in this area.

Trading. Cryptocurrency trading has remained popular for quite a long time, and the world knows dozens of trading platforms with different currencies in the listing and trading instruments. Most exchanges are centralized, i.e. they work according to the classical scheme. Decentralized exchanges or DEX operate without the participation of the administration, i.e. traders enter into a P2P relationship. The funds are not stored in the platform's wallets, which is also very important.

2.3 DeFi and Smart-Contracts

Decentralized finance is inextricably linked to smart contracts. Smart contracts are programs written in Turing-complete, modern, high-level programming languages (Solidity, for example). The main difference of such program code is that after its publication in the blockchain, the contract not only begins its autonomous work, but also loses an ability to edit or change the code. Also it is necessary to say that smart-contracts are not available in all of the blockchain platforms. For example, the most popular platform for developing and deploying such contracts is an Ethereum [2]. The Binance Smart Chain [3] and Matic [4] also provide such an opportunity. But such popular network as the Bitcoin does not provide smart contracts. Smart contracts are used in DeFi, because they help to fully automate the process, save time and avoid paperwork.

2.4 The Main Problem of DeFi

The concept of decentralized finance involves the creation of a single ecosystem in which there are many blockchains that interact with each other. However, each blockchain network was originally created and conceived as an independent, autonomous unit. In our opinion, it is the main problem of DeFi. User's funds have been "locked up" for a very long time. Users do not have an ability to transfer their digital assets to another network, and this is a serious restriction

on the mobility of their funds. In addition, blockchain developers, while choosing one of the platforms, today have to give up the advantages of using other platforms. They could use several blockchains in their project at once, along with their qualities. Instead of this, they should sacrifice such important development indicators as scalability, speed, low fees and so on. That is why the issue of combining and interacting blockchain networks has been relevant for recent years.

3 Related Work

As already mentioned, the problem of combining and interacting blockchains is of interest today. That is why at the moment there are already completed or relatively completed projects in the world which solve it. Let's look at them in more detail.

3.1 Polkadot Ecosystem and Polkadot Projects

Polkadot [5] is a new-generation blockchain protocol that greatly simplifies cross-chain communication and interoperability by bringing multiple blockchains into one network. This network is going to be secured by a GRANDPA consensus algorithm [6], tailored for the Polkadot (a flavor of a Proof of Stake). The most critical parts of the Polkadot network are Relay Chain, Parachains, Parathreads, and Bridges.

Relay chain is the backbone of Polkadot's network and it is the main communication hub between parachains. Validators on this chain are accepting blocks from all parachains and thus provide security for the whole network.

Parachains are independent blockchains that run on top of the Relay Chain and provide chain-specific features to the Polkadot network. Each parachain serves a specialized purpose in the network – think of having a fine-tuned chain for smart contracts, another chain that provides a stable coin for payments between chains or a parachain which brings a decentralized energy industry to the network. Each parachain is maintained by the collator which is responsible for producing chain blocks. Parachains also benefit from a shared security model provided by the Relay Chain so they are already secured against 51% attacks or similar. However, there is only a limited amount of parachains in the network (and the number will be increasing in the future) so there is a system of public auctions where parachain candidates have to compete in order to obtain their own slot.

Parathreads are very similar to parachains from a technical point of view, however, they are very different from an economical standpoint. As we said in the previous paragraph, parachains have to compete in auctions in order to become part of the network. On the other hand, the parathread slot can be leased almost instantly and for only a short period of time. This provides a different way to run projects on the Polkadot – some of those projects can benefit from trying

out the network before purchasing an expensive parachain slot, others can run as a parathreads before they win an auction for a slot.

Bridges are a special kind of parachain. Bridges connect other already running blockchains into the ecosystem (like a BTC or ETH) and allow for transfers of tokens between Polkadot and outside networks.

The peculiarity of the system is that transactions can be carried out simultaneously and distributed between blockchains. The main goal of the Polkadot ecosystem is to make sure that all participating blockchains remain secure and transactions are carried out in good faith.

The two issues most blockchain-based systems need to solve are scalability: the number of transactions per second the network can handle, and governance: how the community manages protocol upgrades and changes. Polkadot aims to solve both of these problems.

Many different projects are based on the Polkadot technology. Some of them coincide in their essence and purpose with our goals. For example, Polkaswap. Polkaswap is a decentralized exchange for the Polkadot ecosystem. It provides a framework that allows users to connect multiple blockchains using bridges and become an exchange for connecting Polkadot participants and other blockchains for efficient asset trading. The principle of operation and implementation of this project is that Polkadot simplifies the process of combining many assets from as many chains as possible by providing a Host relay chain, a cross-chain message transfer protocol XCMP and a SPREE module (Shared Protected Runtime Execution Enclaves). In addition to this project, there are several similar ones.

The disadvantages of projects based on Polkadot are as follows. Firstly, the analysis showed that all similar and interesting projects are stuck in the development stage and do not develop further. The second disadvantage is more significant and weighty. It is about security. In projects on Polkadot the exchange process is implemented as follows:

- To make the transfer of tokens from one blockchain network to another, the process of freezing tokens takes place;
- The freezing of tokens implies the transfer by the user of his funds, which he wants to exchange, to the "storage";
- Next, the logic should be implemented: if the tokens come to the storage, then hold them, and an equivalent amount of them should be credited to the specified address in the target network.

The security issue is that the storage is just an address, not a smart-contract. Accordingly, there are no guarantees and protection of users from fraud or technical failure. It can easily happen that the user will send the tokens that will remain in the storage, and the equivalent amount will not come to him.

3.2 Bridges

A blockchain bridge is an interconnected link that provides communication and interaction between two blockchain systems. By connecting two blockchain

networks, blockchain bridges help decentralized applications take advantage of both systems, not just their host platform. For example, an application hosted on Ethereum and linked to the EOS blockchain can use the functionality of Ethereum smart contracts, as well as the scalability of EOS. Thanks to blockchain bridges, any data, information and tokens can be transferred between two blockchain platforms. These bridges are regulated by the mint-and-burn protocol. The token transfer does not take place literally; rather, when a token is needed to transfer from one blockchain to another, it is burned on the first, and the equivalent token is minted on the other.

An example of such bridge is the Panama Bridge [7], a new solution that allows users to transfer their cross-chain assets from centralized or decentralized wallets to the Binance Smart Chain (BSC). Panama Bridge provides an API. This means that we can use the Panama Bridge on our platform to exchange the tokens. We can collect POST or GET requests through the form and send them to the bridge. The disadvantage of this approach is its limitation. The Panama Bridge, like the other bridges, connects only two blockchains (in this case, Ethereum and Binance Smart Chain). Thus, it is not possible to talk about a single ecosystem using only one bridge.

4 The "Wish Swap" Project Idea

The Wish token [8] has the BEP2 format [9] and is placed in the Binance [3] blockchain, which limits its use. It is necessary to develop a mechanism for exchanging tokens:

- BEP2 Wish to Ethereum tokens ERC-20 and Binance Smart Chain tokens BEP-20;
- Ethereum Tokens ERC-20 and Binance Smart Chain Tokens BEP-20 to BEP2 Wish;
- Ethereum ERC-20 Tokens to Binance Smart Chain BEP-20 Tokens;
- Binance Smart Chain BEP-20 Tokens to Ethereum ERC-20 Tokens.

5 Implementation

5.1 Token Smart Contracts

In order to exchange the Wish token, it was decided to implement the contracts of the token analogues in the Binance Smart Chain (BEP20) and Ethereum [2] (ERC-20) networks, with the names BWish and WWish, respectively. All project contracts were developed in the Solidity language. Solidity is a JavaScript-like object-oriented language for developing smart contracts. It is cross-platform, so it was easy to generalize the task of writing tokens to two platforms. The token contracts, in addition to the standard functions and fields, had to contain the functions and events (events reflected in the transaction log) transferToEthereum/transferToBSC (transfer function to the Ethereum network, or

Binance Smart Chain; depending on the network) and transferToBC (transfer of tokens to Binance Chain), as well as the mint and burn functions. These functions allow you to issue/burn tokens and are only available to the contract owner (in our case, the backend) to prevent uncontrolled issue/burning of tokens.

In addition to token contracts, a Python backend and scanner were implemented.

5.2 Exchange Process with Binance Chain Network

In the Binance Chain network was created an address for exchange with the Ethereum and Binance Smart Chain blockchains. The address is a "swap contract", and the scanner has to catch transfers to it. Transactions on the Binance network contain a Memo field, which may contain additional information. A user who wants to exchange Wish tokens from the Binance Chain network, via the frontend (Django+React), or via binance.org, or via Binance Chain Wallet sends its BEP2 Wish tokens to this address, filling in the memo field according to the rules. The memo field must contain the name of the network to which the user wants to transfer tokens, as well as their number. The scanner scans operations with the BEP2 Wish token. When a transaction is detected to the exchange address, it sends it to the backend via RabbitMQ. The backend accesses the BEP20 BWish token contract and issues tokens to the user's address minus the set commission (a commission is provided for the transfer in the target network tokens). Tokens in the Binance network cannot be destroyed and they remain on the exchange address, only the backend has access to sending from the exchange address, so the number of tokens on the exchange address corresponds to the number of tokens in the Binance Smart Chain and Ethereum networks.

5.3 Exchange Process with Binance Smart Chain and Ethereum Networks

The exchange of tokens carried out from these networks is almost similar. The difference is that:

- here users interact not with the address, but with the smart contract;
- when tokens are transferred to the contract and successfully credited to the target account, they are burned.

The Exchange Process from Binance Smart Chain to Binance. A smart contract has been created in the Binance Smart Chain network for exchange with other blockchains. The procedure for exchanging **BEP 20 BWish** tokens for tokens in **the Binance Chain** network will be as follows:

- the user specifies the parameters on the page:
  1. the network to which the tokens changes (Binance Chain);
  2. receiver address;
  3. the amount of exchanged tokens.

- the user calls the transferToBC function of the BWish contract (the frontend forms a transaction);
- when calling the transaction function, the BEP20 tokens are burned;
- the scanner catches the event of the BWish contract and sends it to the backend;
- the backend makes a Wish transfer to the user's address in the Binance Chain

Only the backend has the right to access the token sending function, which protects against uncontrolled token issuance. All operations with the token are performed within a single transaction, if it was successful, the tokens are burned.

The Exchange Process from Ethereum to Binance Chain. For the Ethereum network, the exchange process is similar to the exchange from Binance Smart Chain to Binance Chain. A smart contract has been created in the Ethereum network for exchange with other blockchains. The procedure for exchanging **ERC 20 WWish** tokens for tokens in **the Binance Chain** network will be as follows:

- the user specifies the parameters on the page:
  1. the network to which the tokens are exchanged (Binance Chain);
  2. receiver address;
  3. the amount of exchanged tokens.
- the user calls the transferToBC function of the WWish contract (the frontend forms a transaction);
- when calling the transaction function, the ERC20 tokens are burned;
- the scanner catches the event of the WWish contract and sends it to the backend;
- the back makes a Wish transfer to the user's address in the Binance Chain

Only the backend has the right to access the token sending function, which protects against uncontrolled token issuance. All operations with the token are performed within a single transaction, if it was successful, the tokens are burned.

The Exchange Process Between Ethereum and Binance Smart Chain Networks. The procedure for exchanging BEP 20 BWish tokens for tokens in the Ethereum network will be as follows:

- the user specifies the following parameters:
  1. the network to which the tokens are exchanged (Binance Chain);
  2. receiver address;
  3. the amount of exchanged tokens.
- the user calls the transferToEthereum function of the BWish contract or transferToBSC function of the WWish contract(the frontend forms a transaction);
- when calling the transaction function, the BEP20 or ERC20 tokens are burned;

– the scanner catches an event on the BWish or WWish contract and sends it to the backend via RabbitMQ;
– the backend address call to WWish token or BWish token to mint tokens to the user's address in current network.

Similarly, only the backend has the right to access the token sending function, which protects against uncontrolled token issuance and all operations with the token are performed within a single transaction. If it was successful, the tokens are burned.

5.4 Project Architecture

The final architecture of the project is shown in Fig. 1:

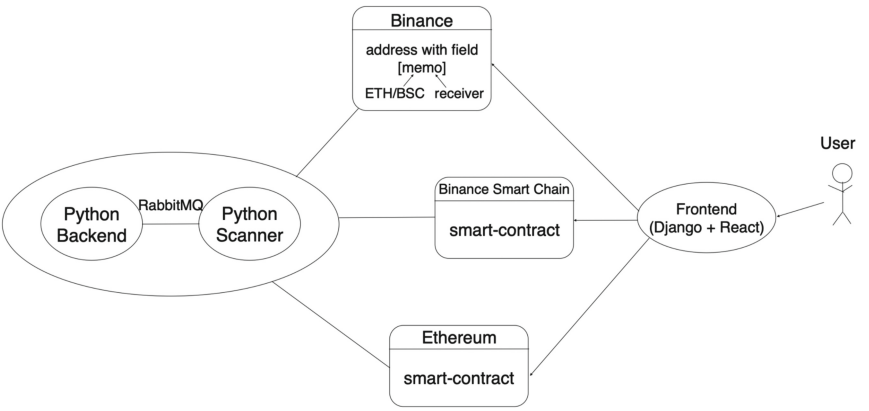

Fig. 1. Project architecture.

The Solidity Token smart contract template includes the following features:

– transferToBC and transferToBSC/transferToEthereum functions (depending on the network) - transfer of tokens in the Binance Chain and Binance Smart Chain/Ethereum networks;
– mint function - a function that is available only to the contract owner, in our case, it is the backend; mint function is necessary to charge tokens to a specific address;
– burn function - it is also used by the backend in the Ethereum and Binance Smart Chain networks after the transfer function; it burns tokens;
– other ERC-20/BEP-20 standart functions.

The Django + React web application performs the following tasks:

- collecting information from the user;
- generating transactions;
- calling contract functions (mint, transfers, burn);
- calculating transaction fees.

Ethereum and Binance Smart Chain Network Scanners and Binance Chain Network Scanner:

It is important to note that the scanners for the Ethereum and Binance Smart Chain networks are identical, they are configured to check a specific token, token contract, and event. That is, in order for our application to be able to see the call of the functions of our contracts coming from the frontend, we just need to use the API provided by Etherscan [10] (The Ethereum Blockchain Explorer) or BscScan [11] (Binance Smart Chain Explorer) (block-observers). And the scanner for the Binance Chain network works in a unique way due to the high frequency of transactions in the Binance Chain. Instead of parsing all blocks in a row, a list of transactions involving the address for the last day is requested with some frequency. When you restart the first request, the data for the week is returned. Network registry scanners scan all transactions in an open registry to send confirmation messages to the backend via a queue (RabbitMQ).

5.5 Security

The current backend has the following architecture: there are N networks/blockchains in which the exchange takes place, each of them has a contract/address that can mint tokens to the addresses of end users. Such a contract/address has an owner who has access, and accordingly, the owner has a private key for such operations.

In fact, we have a small number of private keys (for example, 3 contracts = 3 keys), which in some way makes the task easier, unlike if there were an ever-growing set of private keys needed.

It is not a good idea to store private keys in a database in a pure form. The options for avoiding the problem are as follows.

Refuse to Store the Private Key in Pure Text in the Database Field. There is an option to use Encrypted Field/Symmetric Field/PGP Encrypted Field. Such fields work on the principle that the data stored in the field is encrypted, and cannot be obtained in the absence of the correct password/passphrase/secret key required for decryption (such a key is stored separately, and for example, if you get the database, but do not have such a key, it will be impossible to decrypt the table field).

Advantages of this option: even if someone takes possession of the table, he will not be able to see the keys. Even if you dump the table and pull it out from somewhere, you won't be able to decrypt it either. Also, this practice can be freely combined with other options for strengthening the security of the backend, and it almost never hurts.

Disadvantages of this option: we still have some key on the server/in the settings, but this can not be avoided with other options.

Do Not Store the Key in the Backend Database at All Option 1: It is a good practice to divide the logic of working with the private key and working directly with the rest of the backend into two indirectly dependent components. Since in the current stages and architectures, the backend is already essentially multi-component (scanners, receivers, deferred tasks, web handler), it will not be difficult to make another microservice that would work directly with keys and their receipt/issuance, and keep the rest of the logic in the main part of the backend.

- Such a service should be separated from the backend, but it can be located as a separate subservice in the Docker-Compose backend, or it can be put in a separate application running on its own, perhaps even on a separate server;
- Such a service must communicate with bacon necessarily with encryption, but it can be either an HTTP or a RabbitMQ channel;
- Such a service must have verification of everything that comes to it for the correctness of the sender. The API key or the secret message being transmitted. Perhaps even one-time, according to this scheme:
  1. The main backend requests a secret message from the private backend;
  2. receives the message and encodes it in a certain way;
  3. sends the encoded and initial message to the private backend;
  4. private backend will check whether it is encoded correctly and if everything is in order will give the private key.

Option 2

This option is more suitable for networks similar to the Ethereum network. And for networks like Binance Chain, it is not quite suitable. The idea is that the main backend does not sign anything at all, which means that it will not need private keys. The very same transaction signature would occur on a separate backend/service that accepts the signature parameters, and gives the already signed message:

- The main backend sends parameters to the signature backend;
- The signature backend signs the transaction and gives it to the main backend.
- Such a service must have verification of everything that comes to it for the correctness of the sender. The API key or the secret message being transmitted. Perhaps even one-time, according to this scheme

The provisions on API keys and encrypted requests from the previous version also apply here.

Keep Private Keys Normally, if Everything Else Is Secure. Necessary to understand that to achieve perfection in not storing private keys at all is a little possible technique, but no methods will save you if you make everything

super-secure, but forget any obvious thing that will put everything in question. It is impossible to get a black box that will save all the private keys, because we need to get them back. Anyway, the level of security should be not only in the logic of the backend, but also in everything around it. There are many points here, but the main ones are:

1. Web Server Security:
   - Imperative HTTPS;
   - Imperative firewall, with only port 443 open to the outside, not counting SSH;
   - Strict request policy (e.g. Fail2ban and other request limits);
   - Proxy servers, Cloudflare - required. If the main server is flooded with HTTP Bandwidth attack or DDoS-obviously everything will fall, and no one knows how to use it;
   - Perhaps you should consider Heroku/Kubernetes/Docker Swarm in order to make the so-called scalability for the backend and reduce the number of fail-places (as, for example, only one backend instance).
2. Security of the server:
   - Imperative firewall again;
   - SSH connection ports can be reassigned and it is a good practice to use it. Obviously, half of the Internet uses port 22, why not put another one?
   - Access to the server is NOT by password. The RSA and ECDSA algorithms exist for a reason, and this also needs to be used. No one will get in without a key, unless you will approve it.

Security Implementation. After studying all the above methods and options for protecting private keys, it was decided to implement a combined solution that would contain the maximum number of the listed protection methods.

6 Analysis of the Results Obtained

Comparing the resulting solution with similar projects:

	Wish swap	PolkaSwap	Panama bridge
Security	Yes	No	Yes
Implementation	Yes	In future	Yes
Versatility	No	Yes	No
Scalability	Yes	Yes	No
Fees	100 WWish or 5 Wish/BWish	0.3%	0.001 BNB

For greater clarity, Fig. 2 shows histograms that display quantitative estimates of the performance of the compared services.

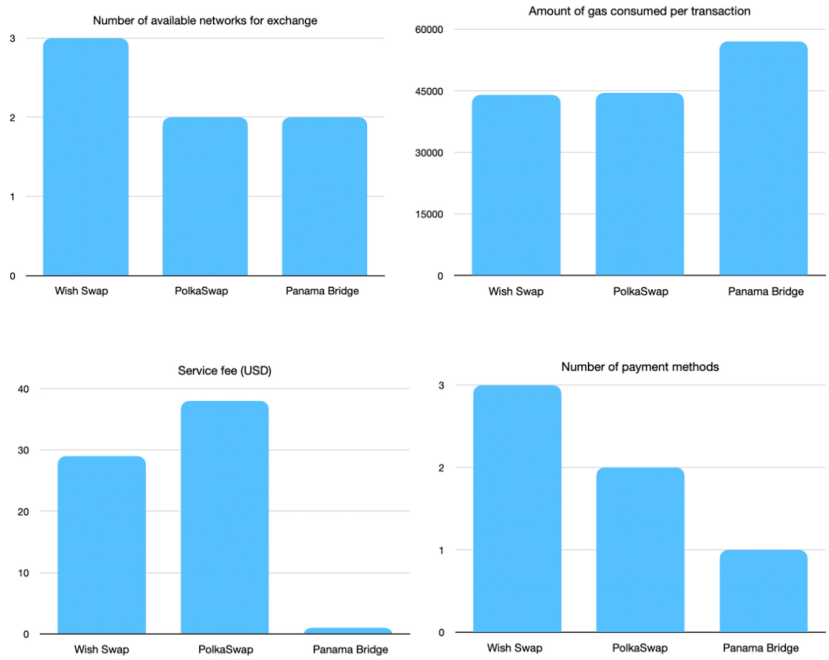

Fig. 2. Comparison wish swap with other projects.

Finally it is clear that our solution meets the security requirements, and also has prospects for development when adding new blockchains, such as, for example, Tron, Neo, Waves, and others.

7 Conclusion

A few years ago, the transfer of tokens and any other information from one network to another was absolutely not possible. However, today, we have proved from our experience that in this direction it is possible and necessary to build useful and completely secure solutions to provide users with as much freedom as possible and remove all possible boundaries in the use of blockchain technology.

The service is planned to be developed further by connecting more and more new networks.

References

1. Zetzsche, D.A., Arner, D.W., Buckley, R.P.: Blockchain disruption and decentralized finance: the rise of decentralized business models. J. Financ. Regul. **6**(2), 172–203 (2020)
2. Ethereum [Electronic resource]. https://ethereum.org (date of the application: 11.03.2021)

3. Binance [Electronic resource]. https://docs.binance.org (date of the application: 11.03.2021)
4. Matic [Electronic resource]. https://matic.network/ (date of the application: 11.03.2021)
5. Polkadot [Electronic resource]. https://wiki.polkadot.network/docs (date of the application: 11.03.2021)
6. Stewart, A., Kokoris-Kogia, E.: GRANDPA: a Byzantie finality gadget (2020)
7. Official Binance Panama Bridge Webpage. https://www.binance.org/en/bridge (date of the application: 11.03.2021)
8. Wish BEP2 Token [Electronic resource]. https://explorer.binance.org/asset/WISH-2D5 (date of the application: 11.03.2021)
9. The definition of BEP2 token standard by Binance Academy [Electronic resource]. https://academy.binance.com/en/glossary/bep-2 (date of the application: 11.03.2021)
10. Etherscan (Ethereum Explorer) [Electronic resource]. https://etherscan.io/ (date of the application: 11.03.2021)
11. BscScan (Binance Smart Chain Explorer) [Electronic resource]. https://bscscan.com/ (date of the application: 11.03.2021)