An Approach to Transaction Delegation in Self-protected Decentralized Data Platforms

© Mykyta Savchenko[0000-0003-1107-0461], © Vitaliy Tsyganok[0000-0002-0821-4877], and © Oleh Andriichuk[0000-0003-2569-2026]

Institute for Information Recording of National Academy of Sciences of Ukraine, Kyiv, Ukraine
zitros.lab@gmail.com, tsyganok@ipri.kiev.ua, andriichuk@ipri.kiev.ua

Abstract. This article presents an overview of existing approaches to transaction delegation in self-protected decentralized data platforms (distributed ledger technology-based decentralized networks), using the example of the most known projects of the Ethereum decentralized data platforms. The four most commonly used approaches are considered, with their advantages and disadvantages highlighted, and the limitations of solutions based on them being outlined. A universal approach to transaction delegation in self-protected decentralized data platforms is developed, which does not require decentralized programs to be standardized, as well as the dedicated server-side processing logic. In addition, auxiliary client and server-side programs are provided, which can be applied to any decentralized programs. Outlined the possible vector of application of the developed approach to decision support systems.

Keywords: Distributed ledger technology, decentralized data platforms, delegated transactions, Ethereum, decision support systems.

Introduction

Followed by the modern trends in information technology, the development of self-protected decentralized data platforms and DLT (Distributed Ledger Technology) is gaining more and more attention, which is predicted to play an enormous role in almost every area in the future: financial, government, healthcare, trading, etc. Such decentralized platforms in 2019 are mainly represented by blockchain technology, as well as other technologies that provide similar characteristics for systems built on top of them.

A self-protected decentralized data platform is standardized storage, security, support, processing and data management system which is maintained by many automated computing nodes, forming a single dedicated network resistant to any unauthorized changes or attacks. Distributed Ledger Technology (DLT) is typically provided as a basis for such data platforms. The main feature of such data platforms is the resistance to any attack on the data and unwanted impact on the decentralized network itself. This makes it virtually impossible to tamper with, to delete or make data unavailable. The

main characteristics of self-protected decentralized data platforms are their bandwidth (usually the number of transactions per unit of time), scalability (ability to scale according to the bandwidth growth), degree of decentralization (full or partial), cost of maintenance (computational and monetary), and others.

Examples of such decentralized data platforms are Bitcoin, Ethereum, EOS, IOTA, Hedera Hashgraph, Holochain, Radix, Telegram Open Network (TON) projects and more. Each platform supports its own dedicated network using tens of hundreds of thousands of computational nodes located all around the globe. Self-protected decentralized data platforms are already used today for a huge number of tasks: creating programmable cryptocurrencies, to secure data registers, such as, for example, a cadastral registry or a register of public documents, voting, registering cargo or goods, tokenization of real-world objects, etc.

The relatively new but increasingly relevant technologies which underpin self-protected decentralized data platforms are evolving as the data platforms themselves. The very first self-protected decentralized Bitcoin data platform, which allows for only 7 transactions per second, has existed for over 10 years and has not yet been successfully attacked, excluding some external cases of big thefts that were out of the technology scope. This level of protection and trust pushes an upgrade of all classic software systems to self-protected decentralized data platforms. But there are a number of disadvantages that make it impossible to do so at this time. At first, none of the decentralized platforms currently available are scalable and therefore cannot handle a large number of transactions and requests. Secondly, these platforms require a specific approach to interact with them, which is often too complex or too costly for end users. Often, applications that are developed on top of the particular decentralized platform simplify their solutions, thus losing the data protection property that the platform could provide.

Therefore, one of the most important problems of using such data platforms is the complexity, as well as the lack of a conventional or universal approach to building decentralized applications in order to solve the complexity problem for the end user. This paper describes a universal approach to building such applications, regardless of the platform used. In addition, the advantages and disadvantages of using this approach in real-world applications are compared and recommendations for its use in decision support systems are given.

1 The Problem of Paying Multi-Currency Fees in a Secure Decentralized Data Platform

None of the existing and theoretically possible decentralized networks and self-protected data platforms that form such networks can exist without their continued support by all network members. In addition, each participant must be sufficiently motivated to remain on the network and perform the functions of ensuring its integrity and security [1]. There are several types of incentives for the network members to apply in today's decentralized data platforms:

– Financial Reward. It is currently the major incentive and it is used in almost any

decentralized data platform: Bitcoin, Ethereum, EOS. Network members who support a particular decentralized data platform receive a reward in the form of a specific asset used in the network, in which network users pay fees. Also, the decentralized platform algorithm may provide for predictable inflation, which deterministically generates a certain amount of that asset (for example, an additional 1% per year), distributing it among participants who support the network.

- Networking privileges. For example, an ability to perform more transactions in it in exchange for supporting the network. This incentive is used in platforms such as IOTA. By design of their decentralized data platform, users of the network themselves or the companies that need to use the network are obliged to perform the support functions of the network to use it.
- Volunteering or agreement. The volunteerism principle is used (or was used temporarily, for example, when starting a new network) in decentralized networks such as EOS, IOTA. That is, the network is supported by volunteers who mostly interested in proving themselves to the community and gaining a certain rating in the network. While volunteerism or agreements are largely used in private networks, there are more and more examples now of using this approach in the public networks, where several institutions agree to support a shared network and build a self-protected decentralized data platform on top of it. A prime example of such a platform is the Libra project, started by Facebook, which includes a large number of organizations such as Visa, Mastercard, PayPal, Uber and others.
- Combinations of the above incentives.

However, almost every modern decentralized data platform use financial rewards as an incentive for those who support the network. Also, network users pay fees to execute their transactions in the network This commission is needed not only to reward those who support a decentralized data platform but also as a precautionary step against spam attacks on the decentralized network. That is, in order to use the data platform or applications based on it, the user must first purchase a certain amount of cryptocurrency that is used to pay commissions in a particular decentralized network, and only then start using the data platform or the application on top of it.

Acquiring cryptocurrency for each decentralized data platform is not an easy process: in addition to users having to create their own wallet within a decentralized platform, users are also forced to use the so-called exchanges to receive cryptocurrency through which they can interact with the decentralized application. Today, currency exchange services are quite sophisticated, however, there are some limitations to implementing such an exchange, from the user's point of view:

- At first, the online currency exchange is very dependent on the country in which the user is located, as well as its laws. As of today, many people still do not have the ability to buy cryptocurrencies.
- Secondly, because of exchange restrictions, users cannot buy the small amount of cryptocurrency needed for only a few targeted transactions on the network. Therefore, they have to buy a large amount of cryptocurrency (usually not less than $50) at once.

Unfortunately, most Internet users refuse to use web services that have a complicated registration or their usage process. And in the case when internet service is based on decentralized data platforms, the use is largely limited to the userbase which is able to buy a cryptocurrency and effectively go through an app onboarding process. Figure 1.1 shows the initial steps required to prepare you to work with decentralized data platforms.

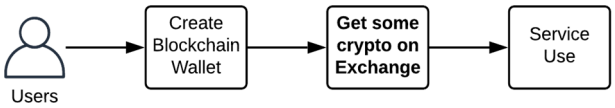

Fig. 1.1. — The diagram of the steps that the user goes through before using decentralized applications

Thus, the use of decentralized applications based on self-protected decentralized data platforms either becomes very narrow or is simplified to systems where the use of the decentralized data platform loses most of its important functions, becoming more centralized and vulnerable to attacks.

2 Approaches to Simplifying Solutions Based on Self-Protected Decentralized Data Platforms

The use of decentralized applications known as DApps or Decentralized Applications requires special software known as crypto wallets. An account (wallet) in a decentralized data platform is the user's unique address used to perform certain actions with the data platform [3]. An account can be generated by the user for free (such as it is in decentralized Ethereum or Bitcoin data platforms) or pre-registered (such as in EOS decentralized data platform). In any case, to create an account, the user generates a secret (private key) that allows them and only them to perform certain actions in a decentralized platform. Special software (crypto wallets) keep the secret (private key) of the user offline and acts as an auxiliary program for interacting with the decentralized platform. Quite often, using just a single secret, you can generate virtually an infinite number of accounts by utilizing the Hierarchical Deterministic (HD) key creation algorithm, such as BIP32 [4], if doing so is supported by a crypto wallet.

Crypto wallets usually exist in the form of mobile applications, physical devices with additional software, plug-ins for Internet browsers, or individual browsers with build-in support of a particular decentralized data platform. Examples of existing crypto wallets are Trezor, Ledger (physical devices) [5], Trust, Metamask (mobile applications), Metamask (extensions for Internet browsers), Mist, Brave, Opera (Internet browsers with built-in crypto wallets) [6].

Creating a wallet is usually an easy task, but it requires additional time and user attention. If the secret is lost, the user will no longer be able to restore access to his account [7]. But the biggest difficulty, as noted above, is to refill the account with a certain amount of cryptocurrency, allowing the user to work with the decentralized ap-

plication. Decentralized application simplifications are typically applied at the cryptocurrency purchase step. Thus, from the initial steps shown in Figure 1.1, there is only one step — creating a crypto wallet and an account for a decentralized data platform. A simplified diagram of the steps required for the user to prepare before using the decentralized application is shown in Figure 2.1.

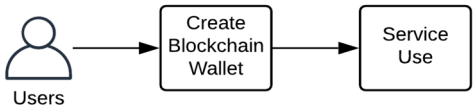

Fig. 2.1. — The diagram of the steps that the user goes through before using decentralized applications without a need to purchase cryptocurrency

This simplification is usually done by utilizing transaction delegation principles — that is, delegating the right to conduct a particular transaction in the network from one account (the user) to another (the delegate) [8, 9]. Thus, the commission for this transaction will be paid by the delegate (the account conducting the transaction), but not by the account that intends to complete the transaction. There is a number of existing approaches to transaction delegation, each having pros and cons. An overview of these approaches and a comparison of their advantages and disadvantages is provided below.

2.1 Trusted Transaction Delegation

The prime example of a trusted transaction delegation is granting a delegate the right to perform any transactions on behalf of some account. Decentralized data platforms such as Ethereum have even developed standards for writing decentralized token programs (eg, ERC777 [10]), which regulate transaction delegation through a trusted transaction delegation approach.

The ERC777 standard, which describes trusted transaction delegation, allows accounts to transfer the right to control their own tokens (programmable cryptocurrencies) to other accounts, as well as to deauthorize particular delegates which were previously granted a right to control tokens. In this way, the use of a decentralized application can be simplified since the transactions to be performed by the user will be carried out by the delegate and not by the user himself. Only one transaction will be left to the user which actually allows the trusted delegate to use tokens on behalf of their account, and any subsequent transactions will be carried out by the delegate. The scheme of a trusted delegated transaction approach is shown in Figure 2.2.

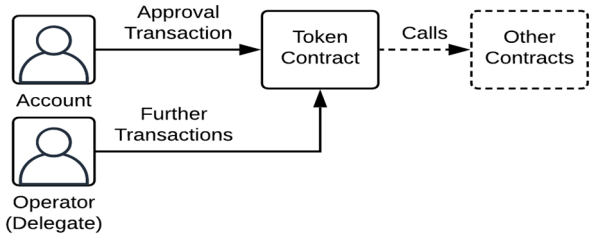

Fig. 2.2. — Trusted Transaction Delegation approach scheme

This approach makes decentralized application easier to use because all user actions can now be performed by a trusted delegate, who also pays transaction fees instead of the user. However, this is only possible after the user completes a transaction that grants delegate permission to some delegate account, and this initial transaction still requires paying fees in a platform-specific cryptocurrency. However, ERC777 standard also provides a way to set the default delegate, such as the account of the tokens issuing entity.

Based on the ERC777 concept described above, the following benefits of trusted transaction delegation can be identified:

— Easy to apply.
— Standardized.

Disadvantages of this approach:

— Requires paying fees in a platform-specific cryptocurrency for the first permission granting transaction, unless the default delegate is set up in the decentralized program.
— Requires paying fees in a platform-specific cryptocurrency to delegate/terminate the delegated authorization.
— This approach does not restrict the delegate in what transactions they can perform and is therefore not secure (due to possible vectors of attack on the delegate).
— It can only be applied to new decentralized programs, or to those where this approach has been foreseen.

2.2 Delegating Transactions with Decentralized Auxiliary Identity Programs

Usually, unlike trusted transaction delegation, systems require a more secure approach which can also be applied to existing decentralized programs (smart contracts). In this case, one can create an intermediate decentralized program, which is called the identity contract. This decentralized program acts as a typical decentralized account but is programmed in a specific way, allowing to perform secure delegated transactions without the need for the owner assigned to this decentralized program to pay for transactions. Using this approach, an account holder is only required to authorize transactions on behalf of his or her personal contract through digital signatures. The approach of transaction delegation with the help of decentralized programs is shown in Figure 2.3.

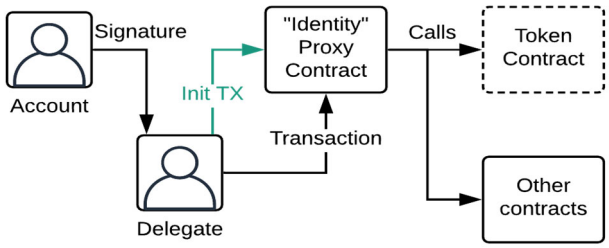

Fig. 2.3. — Transaction Delegation Scheme Using Auxiliary Decentralized Programs

The advantages of this approach include the following:
— It can be applied to any decentralized program.
— This approach is secure because the account holder of the identity contract may have full control over the actions performed by the delegate through digital signatures.

Disadvantages of delegating transaction with decentralized auxiliary identity programs:
— It changes the logic of how decentralized programs should work and introduces the concept of an "identity contract", thus making some existing applications incompatible with it. For example, in this approach, the tokens will own the contract of the account, but not the account itself which is why the balance of tokens in the account's crypto wallet will always be displayed as zero (since the crypto wallet that tokens are stored on the identity contract instead of the user's account).
— Identity contract initialization is required, which potentially opens a new vector for spam attacks on decentralized applications implementing delegated transactions with the use of identity contracts.
— It is hard to standardize.

An example of standards that describe the identity contract with the use of electronic signatures are ERC1056 and ERC780 [11].

2.3 Delegating Transactions without Decentralized Auxiliary Identity Programs

The approach to delegating transactions without the use of intermediate decentralized programs can be classified as a combination of the advantages of the two approaches described above. In this approach, unlike the one which uses decentralized identity programs, all the properties of decentralized identity programs are provided directly in the decentralized program which describes the cryptocurrency. This cryptocurrency (usually token) is used in decentralized application, and its users pay commission in it. And, unlike trusted transaction delegation, the actions that a delegate can perform can be limited and fully controlled by the account itself using digital signatures. The scheme for delegating transactions without the use of auxiliary decentralized programs is shown in Figure 2.4.

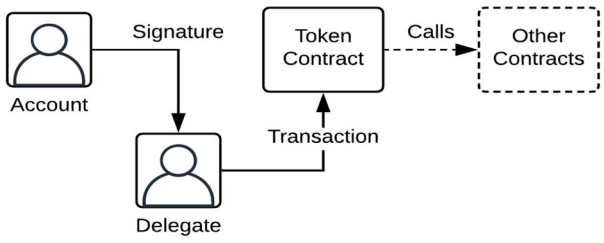

Fig. 2.4. — Transaction delegation scheme without the use of decentralized ancillary programs

The advantage of this approach include:
— No need for initialization transactions, unlike in the previous two approaches.

— This approach is secure because the account holder has full control over the actions that the delegate can perform through digital signatures.
— It does not change the logic of decentralized applications.

The disadvantages of this approach include:
— It is hard to standardize.
— It can be applied either to the new decentralized programs or those which already have some delegated functions in the token decentralized program.

Examples of attempts to standardize this approach include the next proposals: ERC1776, ERC865, ERC1003 and ERC1228 [11, 12], which still remain open as for 2020. The main disadvantage of this approach is that it is difficult to standardize, and this has led to the emergence of many different token standards, which have already been deployed to decentralized networks.

2.4 Delegating Transactions on Decentralized Data Platform Level

Speaking of self-protected decentralized data platforms as a whole, there is a possibility of implementing transaction delegation at the data platform level. However, so far this approach has not become widely adopted and has not been implemented in any of the leading decentralized platforms such as Ethereum, TRON, EOS and others (as of 2020). It is only known that this approach can be implemented in the decentralized platform Ethereum 2.0, which release is planned no earlier than 2022 [13].

An approach to delegating transactions on the level of the data platform itself may appear as the ability to pay a commission for any transaction by any delegate, on some agreed particular terms. For example, a delegate will only conduct the group of account-signed transactions in which at least one transaction will pay the appropriate commission in a particular token, cryptocurrency, or perform any rewarding action for delegate.

This approach requires the support of signed transaction groups at the data platform level. Thus, it is necessary that several signed transactions, where one of the transaction pays a transaction delegation fee, are guaranteed to be executed or rejected in full in case of at least one unsuccessful transaction in the group. The potential implementation scheme of this approach is shown in Figure 2.5.

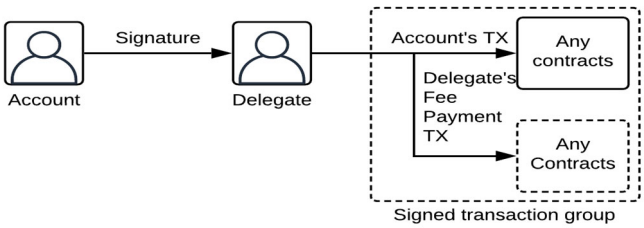

Fig. 2.5. — Transaction delegation scheme at the decentralized data platform level

The advantages of this approach include all the advantages of the approaches described above, excluding these disadvantages:

- As for 2020, this approach is supported by neither of the existing advanced decentralized data platforms.
- It is difficult to standardize at the data platform level. However, the standardization at the level of decentralized applications is possible, as delegates are usually the companies themselves that support the decentralized application and have complete tools to work with their decentralized token programs, including trading and cryptocurrency exchange tools. As for the data platform itself, the problem of paying fees in any arbitrary cryptocurrency or token has a complex nature of the exchange, which does not allow you to accurately estimate the fee to be charged nor to exchange any possible token without centralizing the data platform. As an example, the payment of a commission in token X may be refused by delegate A because it simply does not trust the token or cannot exchange it for another cryptocurrency (no exchangers, prohibited by law, etc).

3 Universal Approach to Transaction Delegation

All of the above methods of transaction delegation have their advantages and disadvantages, and there is no such method that stands out. However, to understand which method is the most applicable and can be enhanced, a review of the most common decentralized applications of 2020 was conducted. The review is based on different transaction delegation approaches used in the Ethereum platform (mostly projects that have successfully completed ICO (Initial Coin Offering): Binance, DreamTeam, Loom Network, Stratis, Maker DAO, OmiseGo, Basic Attention Token, 0x, Golem). Among them are the following trends of decentralized applications that are currently relevant:

- Almost every decentralized application uses its own cryptocurrency token to pay for the use of decentralized or partially decentralized services [14].
- Decentralized applications are intended for a narrow, knowledgeable audience and mostly don't use delegated transactions to facilitate the use of services (to pay for services in a token and additionally require fees in the currency of a decentralized data platform) [15]. Non-common, usually private decentralized applications, which use transaction delegation, have very specialized systems to support only certain decentralized programs.
- As a consequence of the previous point, there is no need for an open ecosystem (market) for delegated transactions. Delegated accounts used in decentralized applications are generally supported by developers of the decentralized application and are of no financial interest to other delegate organizations.
- Transaction delegation is highly centralized, which also follows from the previous point.
- There is a lack of a common standard or approach to building decentralized applications that use transaction delegation.

Thus, it can be concluded that transaction delegation approaches in which delegated functions are embedded in the decentralized token program are most appropriate among existing decentralized programs as each organization tries to develop and support its own transaction delegation ecosystems. But the lack of a single standard or approach

to building decentralized programs or applications that use transaction delegation still remains unresolved.

The universal approach to the transaction delegation described in this article offers a solution to the problem of standardization by writing decentralized programs and their service parts in such a way that there is no need for a single standard at the level of decentralized data platforms. In other words, this approach allows the use of a single service system for any decentralized programs with support for transaction delegation, including already existing decentralized programs, regardless of their implementation. In the case of all of the approaches described above, the transaction delegation was impossible or it is required to build special service systems separately for each implementation.

The universal approach to writing decentralized applications described in this article consists of the following parts:

- An approach to writing a decentralized program, in which the functions of such a program are not limited and can implement any standard of transaction delegation at the choice of the developer.
- The developed back-end server for transaction delegation which is suitable for any decentralized application that uses any approach to transaction delegation.
- A custom client-side user interface in the form of a built-in web widget, which is configurable and can be used as a graphical user interface for delegated transactions.

3.1 The Approach to Creating a Decentralized Program

The universal approach to transaction delegation is based on an approach that does not use intermediate (identity) decentralized programs. That is, this approach extends the one in which delegated functions are embedded directly to a decentralized token program, which is also used to pay commissions. Figure 3.1 below shows a diagram of a user's interaction with the decentralized program without using intermediate decentralized programs.

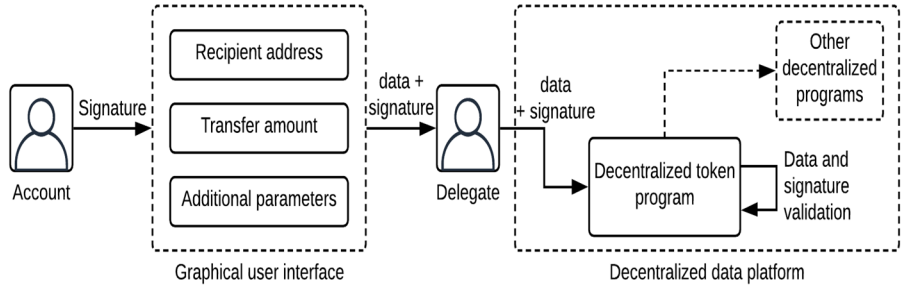

Fig. 3.1. — User interaction with the decentralized program using a transaction delegation approach without intermediate (identity) decentralized programs

The diagram shows the following steps for executing a delegated transaction:

- The user receives prepared data or inputs new data in the graphical user interface.

- The user signs the data entered and the additional parameters exclusively for executing the delegated transaction (such as commission, deadline, transaction ID, etc.) by a digital signature.
- The user transmits the signature and data to the delegate.
- A delegate sends the actual transaction to a decentralized data platform, paying a commission in a platform-specific cryptocurrency.
- The decentralized token program checks the signature and transmitted data to ensure that it is genuine and conducts the transaction.

A decentralized token program can be written using any existing standard: for example, ERC20 or ERC721 [16]. But it should be noted that in order to be able to apply the universal approach described below, it must satisfy the following:

- For each public function of a decentralized program, a similar additional "delegated function" should be provided that will perform the same action but will not depend on the calling account (delegate) and use the digital signature instead to recognize the calling account. In other words, for example, if the decentralized token program has a *transfer* function that allows you to transfer cryptocurrency from one account to another, the *transferViaSignature* function must be additionally provided, which will allow the account to transfer cryptocurrency with the help of a delegate. Alternatively, the contract may provide a single entry point function to call all other functions with the use of electronic signatures (delegate function).
- If possible, the delegated function should support several existing signature standards. Using multiple standards instead of one will increase the chances of an uninterrupted workflow for a decentralized application in the future because over time some signature standards may be declared as deprecated.
- The delegated function must provide a number of additional parameters required to limit its arbitrary use and provide additional security guarantees for the signing account. The recommended parameters are transaction ID, transaction deadline, fee receiving account. All additional parameters must be included in the digital signature and verified by the decentralized program.
- In addition to the points above, the proxy-call function to other decentralized programs may be provided and the corresponding delegated function for it, usually named *approveAndCall*. This will create a complete framework for other decentralized programs to access tokens in a single transaction.

Below is the description of a recommended approach to writing delegated functions on the example of a decentralized Ethereum data platform, namely on the example of a decentralized token program using the ERC20 standard and the transfer function provided by this standard. This approach simplifies the use of a decentralized application by eliminating the additional cryptocurrency that must be paid as a fee in almost any decentralized data platform. It should be noted that this approach can be applied to any self-protected decentralized data platform unless the platform itself provides for another option for delegated transactions.

The ERC20 *transfer* function is intended for transferring a certain number of tokens (programmable cryptocurrency) from the user's account to another account. We suppose that the *transferViaSignature* function is implemented, which will do the same

task as the *transfer* function does, but allows it to be delegated to any other account. That is, unlike the *transfer* function, any account can make a delegated call to the *transferViaSignature* function, and its result will be similar to what the transfer function does, but with respect to the account which made a signature instead of the calling account. Figure 3.2 shows how the function call is performed by the user and the transfer of tokens is made in the case of *transfer* and *transferViaSignature* functions. It should be noted that the *transferViaSignature* function implements the optional commission pay to cover the delegate's account expenses.

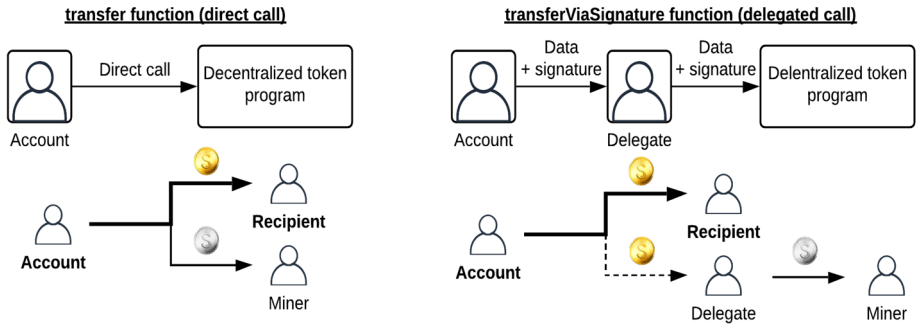

Fig. 3.2. — The scheme of calls to *transfer* and *transferViaSignature* functions and function results

As can be seen from the diagram shown in Figure 3.2, in a delegated call the account pays the transaction fee in the token (cryptocurrency) itself, while the commission required by the self-protected decentralized data platform is paid by the delegate. The delegate then exchanges the token to another asset in background, which covers their expenses. Mainly, this eliminates the need for the account to hold multiple cryptocurrencies (in the case of the Ethereum decentralized data platform, Ether cryptocurrency). It is worth noting that the commission may not be paid at all, but it is shown in the diagram as an example of a balanced economic model where the delegate receives at least something to cover their expenses of conducting the delegated transaction.

The transferViaSignature function not only accepts the same parameters as the transfer function but also has a number of additional parameters that guarantee security in any edge case scenarios. Figure 3.3 shows the recommended function parameters to be implemented in a decentralized token program.

```
contract ERC20 {
  function transfer(to, value) external;
  function transferViaSignature(from, to, value, fee, feeRecipient, deadline, sigId, sig, sigStd) external;
}
```

Fig. 3.3. — Recommended parameters of the functions of a decentralized token program

While the transfer function accepts only two parameters, to (recipient) and value (transfer amount), transferViaSignature additionally accepts the following parameters:

— from — sender account (optional). Unlike the transfer function, the sender account is not the account that sent the transaction, so it may be additionally provided or

retrieved from an electronic-digital signature in-place in the most decentralized data platforms.

- fee — the amount of commission paid by the sender account to the feeRecipient.
- feeRecipient — the address of the account which receives the commission paid by the sender account. It is not recommended to remove this parameter or use the calling address as the delegate's account because typically, a transaction can be cloned and executed faster than the original one (race condition), making the original transaction obsolete.
- deadline — the execution deadline for a delegated transaction, validated in the decentralized program. With this parameter, in case of the loss of signed data or signature for whatever reason, the user can be sure that no one will be able to execute his transaction after some moment and re-sign data once the previous signature expires. Alternatively, they can sign the data under the same sigId.
- sigId is an additional unique transaction identifier, which is validated for its uniqueness in the decentralized program (an optional parameter). In case the signature was lost and the transaction didn't happen, the user can repeat the transaction by using the same sigId to be sure that either the previous transaction happens or the current one but not both.
- sig — a digital signature made by the sender's account, which has all parameters of this function signed, thereby certifying its specific action that the sender account intends to perform.
- sigStd — the signature standard identifier for the decentralized program in case of it using multiple signature standards (recommended).

Therefore, these additional parameters, their signature and their verification on a decentralized program's level limit the delegate's ability to manipulate the delegated transaction in any way. Thus, the user can be sure that the transaction will or won't be executed and its action is clearly defined.

It is worth noting that the introduction of transaction delegation to the system is almost always a partial centralization of a decentralized application since only certain institutions provide services for transaction delegation. Users have to rely on these institutions in order to use transaction delegation. But this centralization should be threatened for the sole purpose of simplifying a decentralized application for the user. In order for decentralized applications to remain decentralized, this approach encourages to write delegated functions only as auxiliary ones, and always keep the original function which allows performing the same action. Alternatively, by having delegated function, in case the delegate doesn't accept the user's transaction, the user should become a delegate for themselves, even though they would need some decentralized platform-related cryptocurrency.

3.2 The Approach to Creating a Transaction Delegation Support Service

In addition to writing the decentralized program that will support the execution of delegated transactions, it is important to create an automated workflow for delegates to conduct transactions in a decentralized data platform. Thus, the task can be described

as creating an automated server-side part that would perform the following functions:
- Supporting the execution of delegated transactions in real-time, including ensuring that the transaction is sent to the network and end up being stored on the network (being mined).
- Collecting all the required data for delegated transactions.
- Calculating the reasonable transaction fee, according to the network load (decentralized network fee) and assets current exchange rates.
- Validating and verifying the user's data.
- Availability of delegated transaction tracking from the user's side.

The task was set to develop such an auxiliary server part, which would perform all the above functions, and would be suitable for any system from decentralized data platforms and decentralized programs, including existing ones which support transaction delegation by any standard. This technical problem is non-trivial as it is necessary to anticipate the following:
- Protection of the centralized system against any attacks: spam attacks, duplication of delegated transactions, direct hacking and delegate secrets steal, race conditions, etc.
- Sequential execution of delegated transactions while delegation requests can be parallel (for some decentralized data platforms).
- Definition and management of transaction parameters (you must always take a profitable transaction fee, count it relatively to the price of the token on the exchange, etc).
- Serving many user requests at the same time.
- The versatility of the back-end server system and the ability to use it for any decentralized programs that support transaction delegation.

Let's take a look at the automatic work that an auxiliary server system should perform on the example of a decentralized data platform Ethereum. The interaction of system components of the back end support system is shown in Figure 3.4.

The auxiliary system for transaction delegation includes the operation of the five components indicated by the Latin letters in Figure 3.4. There are the following components:

A. The client (graphical user interface) — the graphical interface with which the user interacts.
B. The automated server which delegates transactions. There can be several delegation servers, and the client will pick the one which suggests the best rate for transaction delegation.
C. Delegate account or crypto wallet/system that stores the private key and signs the transactions.
D. A decentralized network supported by a self-protected decentralized data platform (Ethereum).
E. The possible exchange module for automatic cryptocurrency exchange transactions by the transaction delegation server to purchase the cryptocurrency needed to pay commissions in a decentralized data platform in the platform-specific cryptocurrency.

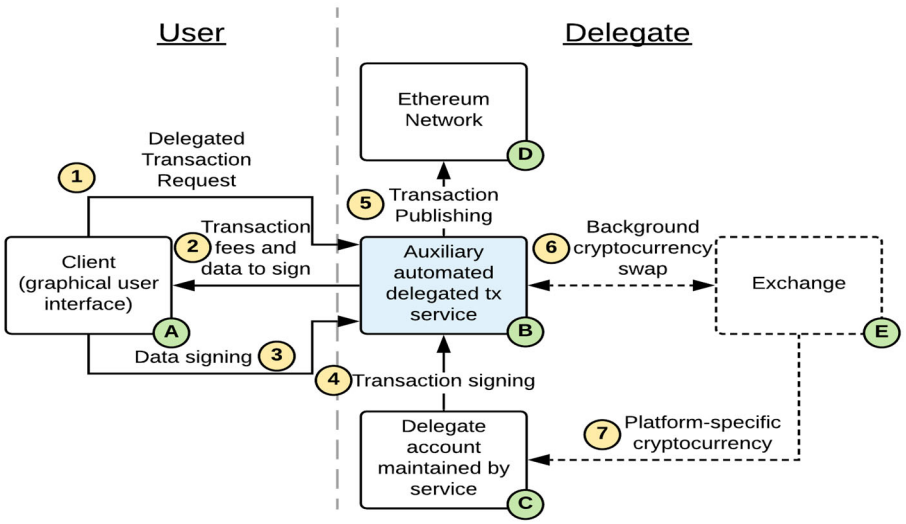

Fig. 3.4. — The interaction between the delegation system components with the auxiliary server part. The numbers indicate the order in which the system components interact.

It should be noted that the auxiliary delegated transactions support application may provide different configurations, for example, with no exchange module, unless required by a specific system (for example, for private delegation of transactions, where no fee is charged to users hence the exchange is not needed). Public transaction delegation is different from private or limited by the fact that the transaction delegation service is in unrestricted public access, and for each transaction the delegate (auxiliary server part) needs to charge at least something from the user in order to cover the costs of submitting transactions to a decentralized network. The delegate can also take a higher fee to get a profit from transaction delegation. The commission (or its part) will be used by the delegate to exchange it for the platform-specific cryptocurrency, which will be paid for the transaction in the decentralized network.

3.3 User's Interaction with the System

Considering the order of the user's interaction with the system and the order of interaction between the system's components, let's review why exactly this order should be used in the public transaction delegation systems. Also, it can be adjusted and simplified in order to be used in private or restricted systems but the framework remains the same, which makes this delegation approach universal. The order of interaction between the system components is also shown in Figure 3.4, and the sequence diagram of the user's interaction with the delegate as well as the delegate's interaction with the decentralized network is shown in Figure 3.5.

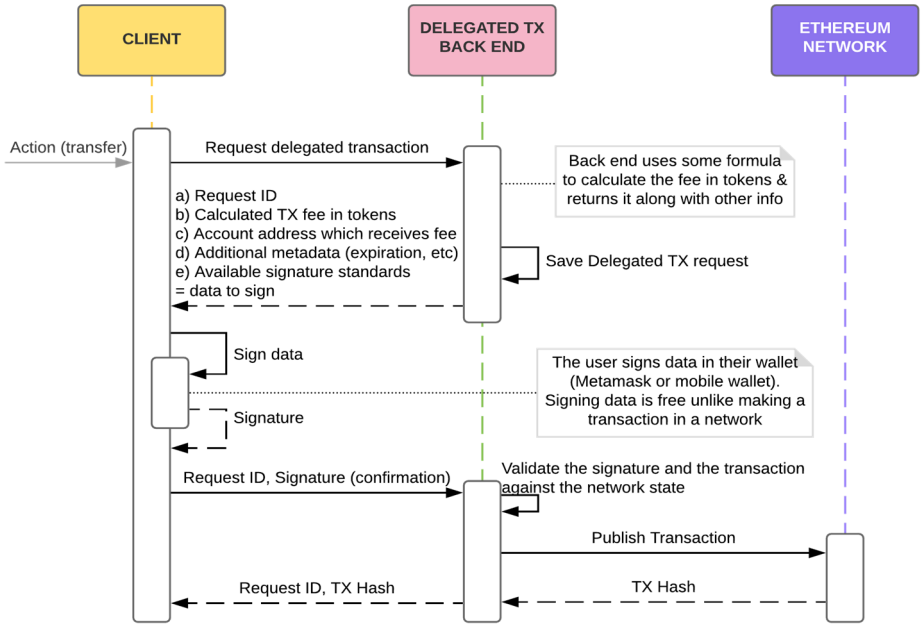

Fig. 3.5. — The diagram demonstrating the sequence of the user's interaction with the delegate and the decentralized network

User interaction with the delegate involves the following automated steps:

1. Delegated transaction request. After the user determines what specific action is intended to be performed, the client prepares the transaction parameters. First of all, it must obtain approval from the Transaction Delegation Support Service that this transaction can be conducted.
2. Delegate's response. Using the developed protocol over the REST API, the client makes such a request, and the delegate in response informs the client about:
   a. The possibility or inability to execute a particular transaction.
   b. The exact fee to be paid by the user in particular token (instead of platform-specific cryptocurrency). In this case, the delegate calculates it automatically, using a prepared algorithm for calculating the cost of commission.
   c. The data to be signed by the user is provided according to the decentralized program. It should be noted that if several signature standards are implemented in the decentralized program, the delegate returns multiple signature options and the client can choose the one that suits him best.
   d. Additional information (delegated transaction ID, estimated transaction time, etc).
3. Data signature and transaction confirmation. After receiving the data for the signature and the meta-information, the client decides which signature method to use and signs the data prepared by the delegate (parameters of the delegated function). Additionally, the client can validate that the data they sign performs the specific action.

However, for example, a delegate may ask for a fee that is too high for a delegated transaction, and therefore the client's decision to conduct the transaction may change. In case of mutual consent of the delegate and the client the signed data is sent to the delegate as confirmation of the execution of the delegated transaction.

4. Transaction signature. In turn, the delegate signs the transaction which is sent to a decentralized network. This transaction includes:
   a. The parameters provided by the user for the transaction (from Figure 3.3: for example, for the transfer function it is "*to*" and "*value*").
   b. Additional options for the decentralized program, which were specified by the delegate (from Figure 3.3: *fee, feeRecipient, sigId, deadline*).
   c. Signature of all the parameters by the user (from Figure 3.3: *sigStd, sig*).
   d. Determination of the next transaction sequential number (in some decentralized networks) and the current commission in the network by the given signature of the transaction.
5. Sending a Transaction to a Decentralized Network. Only after all the above steps the transaction is considered valid and will be accepted by the network. After some short period of time, the transaction will be finalized and the delegate will be able to inform the client of its success or error (the client continuously polls the status of the delegated transaction). When sending a confirmation request to the delegate, the client receives either an error or a transaction ID in the decentralized network along with a delegate-specific delegated transaction ID.
6. Getting the result. While waiting for a delegated transaction, the client constantly polls the server for changes in the delegated transaction status, and when the server responds positively (transaction complete) — the delegated transaction is considered to be completed. The client may also avoid depending on the delegate in this situation and check the status of the transaction in the decentralized network using the transaction ID provided by the delegate in response to the confirmation of the delegated transaction.

Thus, when using the suggested transaction delegation approach, almost all user interactions with the system can be automated, except for the required user signature. By making a digital signature the user agrees to the specific terms of the delegation of the transaction that the delegate has issued to him and, at the same time, make the transaction unchangeable, since the delegate cannot forge a digital signature. The user-provided signature is validated in a decentralized program, which allows the further implementation of any actions by the decentralized program.

3.4 The Application of the Universal Approach to Transaction Delegation in DSS

Decision Support Systems (DSS) are systems that take facts and make recommendations to decision-makers based on many factors. Usually, these guidelines are for informational purposes only, but the result which the system provides can be considered as

the main conclusion for many complex tasks that people usually cannot handle on their own (examples: social group behavior research, complex network of facts and relationships, strategic planning, etc) [17, 18].

One of the key features of decision support systems is that it works with experts. They provide all the necessary data for the system [19], on the basis of which the system then draws conclusions and recommendations. The subsystem of the DSS, which is designed to work with experts who are mostly ordinary users is often stores data in unsecured or centralized registers, which increases the risk of their compromise, forgery, unauthorized removal, etc.

Therefore, first of all, it is important to make sure that the system gets reliable data for the input. The credibility of the input data makes the entire decision support system's authority, as well as the authority of the organization that supports it. Therefore, one of the typical and applicable vectors for using the transaction delegation support system developed and described in the article, as well as for the self-protected decentralized platforms as a whole, is the subsystem of working with the expert input data of the decision support system.

Thus, decision support system experts will be offered to create an account in terms of a self-protected decentralized data platform instead of logging in with a pair of logins/passwords, which differs in complexity only by installing additional software for any modern Internet browser. The further workflow with the experts in a decentralized system will not be fundamentally different compared to a centralized one, but the data that experts input will now be protected by the decentralized data platform. This provides the guarantee of data invariance or data modification under strictly defined rules according to the decentralized program. In the future, the data input by experts without any exception will be considered reliable (except for the bribery or compromise cases of the experts themselves). The input data will then be loaded from the decentralized data platform to the decision support system for further processing.

Conclusions

There are many approaches to delegating transactions in self-protected decentralized data platforms, but only some of them meet the security and ease of use requirements. Developed universal transaction delegation approach combines all the benefits of the existing approaches, has a standardized back-end server and a client. At the same time, it does not require decentralized application standardization, which means it can be used for any existing or new systems that aim to ease the use of decentralized applications by implementing a transaction delegation.

The developed approach and the auxiliary software to support transaction delegation in decentralized data platforms will facilitate the further adaptation of decentralized applications in all spheres of life and technology since this approach simplifies both the user experience within any decentralized system and the necessary efforts of developers to implement the transaction delegation system.

The universal approach to transaction delegation was tested publicly on the DreamTeam.gg project, which has around 2000 decentralized application users (the

number of users is approximated based on the number of project's token owners) and deployed on the internet address send-token.dreamteam.gg. This online service is an open-source solution that can be adapted by any other project based on the self-protected decentralized Ethereum data platform, as well as slightly modified to support any other decentralized network, which supports deploying decentralized programs.

References

1. Savchenko M., Tsyganok V., Andriichuk O. Decision Support Systems' Security Model Based on Decentralized Data Platforms / Selected Papers of the XVIII International Scientific and Practical Conference "Information Technologies and Security" (ITS 2018), pp. 199–208 (2018).
2. Mookherjee D. Decentralization, hierarchies, and incentives: A mechanism design perspective. Journal of Economic Literature, 44(2), pp.367-390 (2006).
3. Raval S. Decentralized Applications: Harnessing Bitcoin's Blockchain Technology. O'Reilly Media, Inc. pp.6-7 (2016).
4. Khovratovich D. and Law J. April. BIP32-Ed25519: Hierarchical Deterministic Keys over a Non-linear Keyspace. In 2017 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW) pp. 27-31 (2017).
5. Arapinis M., Gkaniatsou A., Karakostas D. and Kiayias A. A Formal Treatment of Hardware Wallets. IACR Cryptology ePrint Archive, p.34 (2019).
6. Pustišek M. and Kos A. Approaches to front-end iot application development for the ethereum blockchain. Procedia Computer Science, 129, pp.410-419 (2018).
7. Rezaeighaleh H. and Zou C.C. New Secure Approach to Backup Cryptocurrency Wallets. In submission to IEEE Global Communications Conference-Communication & Information Systems Security Symposium (2019).
8. Cho S., Park S.Y. and Lee S.R. Blockchain consensus rule based dynamic blind voting for non-dependency transaction. International Journal of Grid and Distributed Computing, 10(12), pp.93-106 (2017).
9. Ouaddah A., Elkalam A.A., Ouahman A.A. Towards a novel privacy-preserving access control model based on blockchain technology in IoT. In Europe and MENA Cooperation Advances in Information and Communication Technologies. Springer, Cham. pp.523-533 (2017).
10. Mulders M. A comparison between ERC20, ERC223, and ERC777 token standard [Electronic resource]/Michiel Mulders. Mode of access: https://www. cointelligence. com/content/comparison-erc20-erc223-new-ethereum-erc777-token-standard.–Title from the screen.
11. Chavan AB and Rajeswari K. July. Design and Development of Self-sovereign Identity Using Ethereum Blockchain. In International Conference on Sustainable Communication Networks and Application. Springer, Cham. pp. 523-531 (2019).
12. Ludovic Galabro. ERC865: Pay Transfers in Tokens Instead of Gas, in One Transaction #865. https://github.com/ethereum/EIPs/issues/865, [Last accessed Dec 21, 2019].
13. Wels S.J. Guaranteed-TX: The exploration of a guaranteed cross-shard transaction execution protocol for Ethereum 2.0. Master's thesis, University of Twente. (2019).
14. Lee J.Y. A decentralized token economy: How blockchain and cryptocurrency can revolutionize business. Business Horizons, 62(6), pp.773-784 (2019).
15. Iansiti M. and Lakhani K.R. The truth about blockchain. Harvard Business Review, 95(1), pp.118-127 (2017).

16. Kim M., Hilton B., Burks Z. and Reyes J. November. Integrating Blockchain, Smart Contract-Tokens, and IoT to Design a Food Traceability Solution. In 2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), pp. 335-340 (2018).
17. Saaty T.L. Principia Mathematica Decernendi - Mathematical principles of decision-making - Generalization of the Analytic Network Process to neural firing and synthesis. Pittsburg: RWS Publications. (2010).
18. Tsyganok V., Kadenko S., Andriychuk O., Roik P. Usage of multicriteria decision-making support arsenal for strategic planning in environmental protection sphere / Journal of Multi-Criteria Decision Analysis. 24, pp.227-238 (2017).
19. Totsenko V.G., Tsyganok V.V. Method of paired comparisons using feedback with expert / Journal of Automation and Information Sciences. Volume 31, Issue 7-9, pp.86-96 (1999).