Information & Security

Savchenko, V. Tsyganok & O. Andriichuk
vol. 47, no. 1 (2020): 109-121
https://doi.org/10.11610/isij.4707

Published since 1998 ISSN 0861-5160 (print), ISSN 1314-2119 (online)

Research Article

A Cost-Effective Approach to Securing Systems through Partial Decentralization

Nikita Savchenko [a], ***Vitaliy Tsyganok*** [a,b] (✉), ***Oleh Andriichuk*** [a,b]

[a] *Institute for Information Recording of National Academy of Sciences of Ukraine, Kyiv, Ukraine, http://ipri.kiev.ua*

[b] *National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute," Kyiv, Ukraine, https://kpi.ua*

ABSTRACT:

A study has been conducted on the methods of secure computation and storage decentralization, based on expert and decision support systems' needs. It identified the common disadvantages of modern, Distributed Ledger Technology-based (DLT-based) decentralized systems and suggested a solution to it – the universal transaction delegation method. This method eliminates the complexity of a decentralized system from end-users' point of view, which during the study was identified as the main disadvantage of DLT-based systems. This article provides a brief overview of existing approaches to solve the usage complexity problem, based on data from well-known projects built with Ethereum, the leading DLT-based smart contract platform. As a result of the research, we implemented the universal transaction delegation method which does not depend on the signature standard used in the decentralized program. In addition, we present results of an experimental economic analysis of the solution based on real network tests and data as of 2020.

ARTICLE INFO:

RECEIVED: 08 JUNE 2020
REVISED: 06 SEP 2020
ONLINE: 21 SEP 2020

KEYWORDS:

distributed ledger technology, decentralized data platforms, delegated transactions, Ethereum, data platforms

✉ Tel.: +380444542137; E-mail: tsyganok@ipri.kiev.ua

Introduction

In the past decade, the novel Distributed Ledger Technology (DLT) has proven that it can serve as one of the most reliable, secure, and even lightweight compute and storage layers in the world of Internet and Information Technology. Despite DLTs still have some scalability issues, they are being applied to more and more systems which require the outstanding level of security and cryptography.

In 2020, DLT-based solutions have developed up to the level where complete software systems are built with DTL, meaning the complete absence of the traditional server-side technology such as web servers. An example of such systems are new terms, like crypto Tokens or Decentralized Finance applications. They are built on top of what we call Decentralized Data Platforms (DDPs) in this paper, as they not only store the data but also have a light computation capability. The compute layer allows creating advanced decentralized programs on top of the DDP by using its native programming language, which inherits its security properties. The resulting applications, which typically have a web user interface and a decentralized back end are called DApps, or decentralized applications.

The main property of any DDP is the resistance to attacks, which prevents any unwanted impact on the decentralized programs they run, such as downtime or changing the data in a way which was not programmed in the decentralized program. The underlying DDP algorithm, such as Proof of Stake in blockchain, makes the data practically impossible to delete, tamper with or even decrease its availability, as the platform is supported by hundreds of thousands of computational nodes which constantly verify the network state and perform common verification tasks in competition towards getting a reward.[1] Among other characteristics of DDPs, we can identify their scalability (readiness to scale according to the growth of the network usage), bandwidth (throughput or maximum number of transactions per second), degree of decentralization (complete or partial) and the cost of their maintenance (both monetary and computational) as the main properties.

Examples of such DDPs are Bitcoin, Hashgraph, TON, Ethereum and other projects.[2] These DDPs are supported by their own decentralized networks, which consist of thousands of nodes located worldwide. These data platforms are used today for a variety of tasks: creating programmable cryptocurrencies, securing data storage, such as, for instance, a registry of land properties or a public document registry, registering goods or cargo, cryptographically protected elections, real-world assets tokenization.

One of the most important problems in 2020 preventing the adoption of this technology by large-scale businesses or governments is the complexity of its usage, as well as the lack of the universal approach to building decentralized applications, is solving the task of the system complexity for end users.[3] This paper describes the possibility of building the universal method for DApps, regardless of the DDP used by the decentralized program. Also, it describes the cons and pros of using the described approach with different modern systems and the applied model for various systems based on this approach. The suggested approach improves the security of theoretically any computer system, reducing the risk of tampering with its data and manipulations with data entry, while also requires some special setup and integration steps. This approach uses the universal method for transaction delegation, meaning that it can be applied to any decentralized program on top of any DDP with the support of basic cryptography functions such as hashing. And its usage simplifies the end user experience while does not neglect its security or decentralization levels.

The Complexity of Decentralized Systems Usage

As of 2020, there is no known approach to creating a scalable decentralized network which would not require all network nodes to participate in every transaction validation except sharding, which also does not allow to scan indefinitely.[4] Additionally, each decentralized network participant must obtain some profit from participating in the DDP support process, or otherwise the network will not get enough participants.[5]

Most of modern decentralized data platform use cryptocurrency rewards to reward those who support the network, so that they have a strong financial incentive to participate in it. Hence, network users must pay commission to use the decentralized platform.[6] This commission is used to reward parties who support the decentralized network. Paying fees also plays a precautionary role and protects the network from spam attacks, making them much more expensive to perform.

This way, to use any decentralized data platform, the user has to purchase some amount of platform-specific currency which is used to pay fees. Only after obtaining some cryptocurrency, users can start using the DDP and DApps on top of it. Given that purchasing cryptocurrency for each decentralized data platform is a complex process, especially considering local laws and payment methods available, most Internet users and developers just refrain from using and developing web services based on DDPs. This is what we classify as an adoption and a user experience problem. Currently, all well-known DDPs such as Ethereum or EOS require paying fees, which can only be purchased or handed over from one user to another. DDPs trying to avoid the fee problem built on top of other technologies such as IOTA's Tangle are currently not considered as stable and secure as blockchain-based DDPs.[7]

While the problem of fees is related to DDPs in general, for web services integrating with DDPs, their use becomes very much limited to the number of users which are able not only to understand the concept of decentralized networks, but also to buy a cryptocurrency and effectively go through the DApp onboarding flow. Thus, the use of applications and web services basing on decentralized data platforms either gets very narrow in terms of application, or the decentralized part of the application is simplified to a degree when the whole system loses its important functions and becomes more vulnerable to attacks and centralization problems.

Approaches Used to Simplify the User Experience

The use of "DApps", or Decentralized Applications requires a special software or browser extensions known as wallets, which store the private keys of decentralized identities, or accounts (also known as addresses when referring to the public key). The decentralized account (stored in wallet), in terms of the decentralized data platform, is the user's unique identifier in the network, used to perform certain actions within (or even out of) the data platform.[3] In other words, the account is represented by public and private key pairs, generated solely by the users. In any case, to create the account, the user creates the secret (the private key) that allows them to perform certain operations in particular decentralized platform. Sometimes, it can be compatible with multiple platforms or decentralized networks. Wallets always keep the secret (private key) of the user offline and allow to interact with the decentralized platform. Using just a single secret, which becomes more and more common practice, one can generate practically an infinite number of addresses by utilizing the Hierarchical Deterministic (HD) key creation algorithm, such as BIP32.[8]

Wallets are commonly presented in the form of mobile applications, plugins for Internet browsers, physical devices, individual software, or internet browsers with built-in support of a particular decentralized data platform. Good examples of existing crypto wallets are Trezor and Ledger (physical devices),[9] Trust and Metamask (mobile applications), Metamask (extensions for Internet browsers), Mist, Brave and Opera (Internet browsers with built-in crypto support).[10]

Creating a wallet is usually a quick and easy process, but it anyway requires some time and efforts. If the wallet's secret is lost, users will no longer be able to restore their account, forever losing the access to it.[11] But the biggest difficulty for users remains topping up the account balance with the certain amount of cryptocurrency, which allows them start interacting with the application. The simplifications to this complex step are typically applied at the step of getting cryptocurrency. Thus, transforming a number of steps to the single one — creating a crypto wallet and an account. This typically eliminates the step of obtaining cryptocurrency for the user by replacing the execution of transactions (which require cryptocurrency) with delegation, where another party pays for the transaction instead of the first party.

There are several approaches one can take to utilize delegated transactions:

1. Trusted transaction delegation, where no cryptography is used, and transactions are just executed by the trusted party.
2. Delegating transactions with decentralized auxiliary identity programs, where a utility decentralized program represents the intermediary who owns assets and exposes an interface for different parties to control this program.
3. Delegating transactions without decentralized auxiliary identity programs, where the delegation is already supported natively either by the primary decentralized pro-gram or the decentralized data platform itself.

The latter approach is the most preferable one, however, is the most difficult to standardize at the data platform level. 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 crypto-currency exchange tools. From the decentralized data platform view, the problem of paying commission in any arbitrary cryptocurrency or asset has a complex nature of the exchange, which does not allow the accurate estimation of the fee to be charged nor to exchange any possible asset without partial data platform centralization. For instance, the payment of a commission in asset X may be refused by the delegate A because the delegate simply does not trust the asset as they cannot exchange it for yet another currency they need. This can happen when no exchangers are available, the exchange is restricted by law, there is high risk and volatility in the market, the target asset is fresh new, etc.

Universal Unstandardized Decentralization Method

The method of transaction delegation which does not require standardization is based on the method which does not use intermediate (identity) decentralized programs. This method is an improvement of the one where delegated function(s) are programmed directly in the decentralized asset (token) control program, which is also used to pay commissions. Note that one DDP can have multiple assets on top of it, and below we describe the process of creating delegated transactions for a single asset (as multi-asset delegated transactions are not common).

Figure 1 below describes the interaction of the user with the decentralized program by using the universal delegation approach.

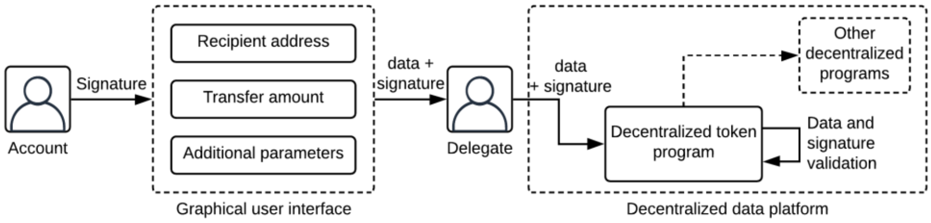


Figure 1: The universal transaction delegation approach demonstrating the user's interaction with the decentralized program.

The diagram below shows the flow of the delegated transaction execution:

- The user gets structured data or types data in the GUI (graphical user interface).
- The user then signs this data with some important additional parameters (such as commission, deadline, transaction ID, etc.) using their decentralized account.

- The user gives the signature along with the signed data to the delegate.
- A delegate sends the expensive transaction to the decentralized data platform, paying a fee in a platform-specific cryptocurrency they already own.
- The decentralized asset program additionally performs checks over the signature and the signed data to ensure that it is genuine before allowing the actual transaction. Thus, the delegate cannot perform any other operations rather than the one signed by the user, i.e. the delegated transaction is 100% specific but is just not specifically determined in time.

A decentralized token program can be developed using any existing standard: for example, ERC721 or ERC20 if the target DDP is Ethereum.[12] It is worth noting that this approach can be easily ported to other blockchain-based DDPs, as many of them are Ethereum Virtual Machine-compatible or have a similar paradigm. To be able to apply the above approach, the following must be implemented in the *expensive asset* (token) decentralized program:

- For every function of a decentralized program which means for the public use, the similar additional “delegated” alternative to this function should be provided that will perform the same action but will not depend on the calling account (delegate), as it is with the default functions. This should be the main concept for designing the asset (token), as it will allow to benefit later. The function should use the provided digital signature to recognize the calling account and act accordingly. For instance, if the decentralized asset program has a “transfer” function that allows to transfer assets from one account to another, the "signedDataTransfer" must be additionally provided, which allows the account to transfer assets with the help of any delegate. Alternatively, the contract can provide the single-entry point function to call any other functions with the use of electronic signatures (delegate function).
- When possible, the delegated function should support several existing signature standards used in the DDP. Using multiple standards instead of one increases chances of the smooth DApp work in the future. It should be noted that over time, some signature standards may become deprecated and using multiple available signature standards makes it more future proof.
- The function itself must take additional arguments required to limit its abnormal use, giving additional security guarantees for the signing account. Some recommended arguments are transaction ID, transaction deadline, and the account which receives the fee. Additional arguments must be added to the signature and to the decentralized program for verification.
- In addition to the above points, such function can even be provided in the dedicated decentralized program, typically utilizing functions similar to Ethereum ERC20s “approveAndCall.” This can create the framework for any programs to even access multiple assets in the single transaction.

Below we provide the guidelines of the recommended approach for creating functions one the example of an Ethereum, using the decentralized asset (token) program utilizing ERC20 standard. The “transfer” function is used as an example; however, any other function can be implemented in a similar way. This approach simplifies the use of a DApp by eliminating the additional cryptocurrency that must be paid as a fee in almost any DDP. It should be noted that this approach can be applied to any DDP unless the platform itself provides for another option for delegated transactions.

The ERC20 transfer function transfers the certain number of assets (the programmable cryptocurrency, or tokens) from one account to another. We suppose that the "signedDataTransfer" function is also present, and does the same what the “transfer” function does, also allowing the delegation of its execution. Unlike “transfer”, “signedDataTransfer” can be invoked by any account not necessarily the one intending to transfer some assets. Its result eventually will be the same to the “transfer.” Figure 2 shows the way of performing the function call by the user and a delegate. The “signedDataTransfer” additionally implements the optional fee payment to cover the delegate's account expenses for performing the original transaction in the decentralized network.

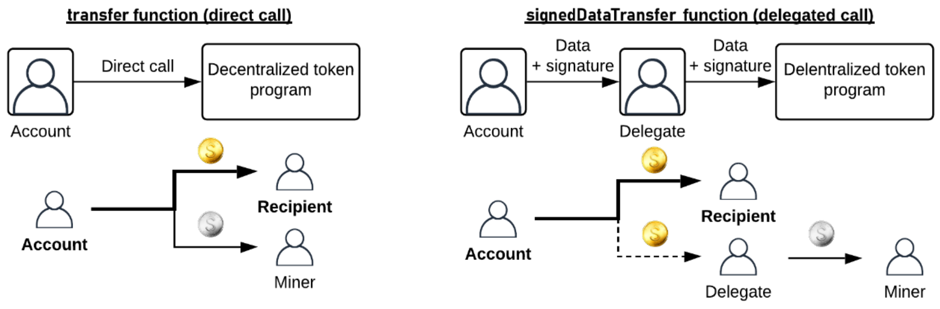
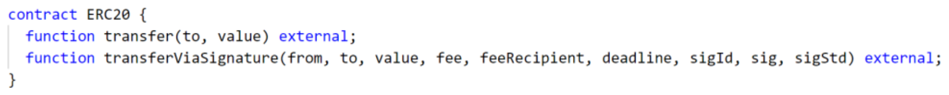

transfer function (direct call)
Direct call
Account
Decentralized token program
Account
Recipient
Miner
signedDataTransfer function (delegated call)
Data + signature
Data + signature
Account
Delegate
Delentralized token program
Account
Recipient
Delegate
Miner

Figure 2: “Transfer” and “signedDataTransfer” functions difference in terms of fee transfers.

As observed from the diagram on the figure 2, within the delegated call the sending account pays the fee in the transferred asset itself, while the commission for this transaction is paid by the delegate. Later, the delegate exchanges the asset to another one, which covers their expenses. Most notably, it eliminates the requirement of account holding multiple currencies (for Ethereum, holding Ether currency). The commission payment can be skipped at all, but it is displayed in the diagram for instance.

The “signedDataTransfer” not only accepts the same arguments as the transfer function but also adds a number of additional parameters. Figure 3 shows the recommended arguments.



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

Figure 3: Recommended arguments of the delegated function for the decentralized asset program.

While the "transfer" accepts just a few arguments, the recipient and transaction value, "signedDataTransfer" additionally accepts more arguments:

- from — sender's address. It can be optional. Unlike in "transfer," the sender's account is not the account that actually publishes the transaction, so it may be additionally provided or retrieved from the signature. This applies to many DDPs.
- fee — the fee in the same asset paid by the sender to the "feeRecipient."
- feeRecipient — an address which receives the commission. It is advised to keep this parameter or use the invoking account as "feeRecipient." Because the transaction sent to the network can be cloned and executed faster than the original one (race condition), which will result in dropping the original transaction.
- deadline — the time when transaction will no longer be possibly. It is validated in the decentralized program. In case of the leak of the signed data or signature, the user can be sure that nobody is be able to execute his transaction or sign the data once the previous signature expires.
- sigId is the unique transaction identifier, validated in the decentralized program. It can be optional when replaced with sequential number. In case when the signature got lost, the signing account can recreate the transaction with the same sigId to ensure that the previous transaction will not ever happen.
- sig — the signature created by the sender's account, with additional arguments signed, thereby confirming the account's intent to perform a specific action.
- sigStd — the identifier for the case of using different signature standards (it is recommended to implement multiple signature standards).

Therefore, these additional arguments and their verification limit the ability to manipulate the transaction in any possible way. The sending account can be sure that the transaction either executed or not, and its action is clearly defined. The only thing which this scheme cannot guarantee is the time of the transaction execution. Delegates may refuse to execute transaction, and the sender will have to either find the other delegate or do the expensive transaction themselves by purchasing cryptocurrency. It is important to note that typically, a delegate is an authority which officially supports and maintains the DApp, hence it is not interested in delaying requests in anyway (but in turn it earns an additional commission for delegating transactions).

Adding transaction delegation to any system almost always results in its partial centralization. Network users must rely on delegating institutions in order to use the application. This centralization is an addition to already decentralized functions (“transfer”) for the sole purpose of system simplification for the end user. For DApps to remain fully decentralized, this approach allows writing delegate capabilities as auxiliary ones, keeping the original functions, which allows performing the same thing, but in the centralized way.

By having the delegated function in case, the delegate does not accept the sending account's transaction, the user themselves can become delegates. This is the only exceptional case when they would need to purchase cryptocurrency in order to transact when delegates are absent.

Cost Estimation

The evaluation operation costs of the expert system were carried out with a preliminary evaluation of the methods of data compactification and based on the actual results of the system operation in the public Ethereum decentralized network (Ethereum mainnet). The study used a system that compacted data into a decentralized network with overwriting obsolete data, but without the use of data grouping. It conducted an average of 2 delegated transactions per day for 3 months in real time, with each transaction changing 768 bits of information in a decentralized registry. Only some transactions were recording new data, while the majority of transactions (90%) were overwriting existing data. The transaction cost, which is set by the user P (gas price) was not minimized by deferred execution of transactions: every transaction was using the current corresponding value of the decentralized network.

The results of the study are aggregated in the Table 1, together with the calculated values for Table 1. The visualization of values throughout the period of running the experiment is shown in Figure 4. The blue graph shows the amount of cryptocurrency Ξ (Ether) on the balance of the account that made the delegated transactions, while orange indicates the number of transactions that occurred on a day.

Where:

- Gas – a unit in Ethereum which measures the absolute cost of performing the transaction in the DDP. The more computation involved in the transaction, the more “gas” is used. Measured in Weis.
- Ether – an expensive asset proportional to the gas spent.

The results obtained prove that the developed transaction delegation system is ready to work in real conditions, despite the fact that at the time of testing it did not implement several significant optimizations that reduce the cost of its operation, namely grouping transactions their deferral to the times of the lowest network utilization (typically at night).

Table 1. Aggregated data derived from the experiment without storage and realtime cost optimizations.

Designation	Value	Unit	Description
t_{start}	1/10/2020	Date	Start of the experiment
t_{end}	3/10/2020	Date	End of the experiment
n	95	Txs	Total number of transactions made
P_{avg}	3.815	GWei	Average cost of gas at the time of transaction
G_{avg}	1.368×10^5	N / A	Average amount of gas consumed for each individual transaction
b_{avg}	768	Bits/Tx	Average amount of data recorded / changed in bits of information
δ_{mar}	128.92	USD	The market price of Ether cryptocurrency as of March 11, 2020
δ_{exp}	5.19	USD	Total spent on the system in real time without additional optimizations
δ_{min}	0.4	USD	Total lowest possible equivalent cost when applying additional optimizations

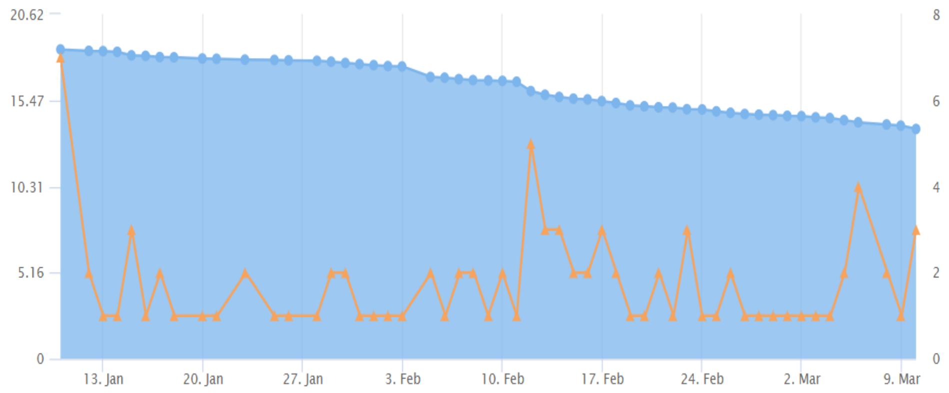


Figure 4: The comparison of spendings on the real system and the number of transactions in the main public network Ethereum without grouping transactions and deferred execution. Orange chart denotes the number of transactions without using grouping (legend on the right), blue chart - transaction costs in $ equivalent (legend on the left).

Based on the results obtained after analyzing different ways of compacting the data, we can determine by how much the cost of system operation is reduced by applying transaction grouping and performing them at the lowest possible network load (such as at night or on weekends). It should be noted again that the optimized data impose additional constraints on the system functions, however, when using the suggested method of transaction delegation, these restrictions are not critical to the most of systems it is applicable for.

Thus, we obtain:

- carrying out transactions during the minimal load of the decentralized network will allow to reach the minimum value of $\mathrm{P_{avg}} = 1.01 \times 10^9$, which reduces the costs by $\mathrm{o_1} = 74\%$ in relation to the results of the conducted study with $\mathrm{P_{avg}}$ obtained in real time;
- when applying the method of compacting data with overwriting outdated information and grouping transactions, $\mathrm{n} = 95$ transactions could be carried out in just two transactions with the total amount of data being written $\mathrm{b} = \mathrm{n} \times \mathrm{b_{avg}}$, with the additional savings of $\mathrm{o_2} = 74\%$. This value can be calculated by substituting the aggregate values in formula (2) for $\mathrm{g_0} = \mathrm{g_{0\ extra\ min}} = 51000$;
- when applying both optimizations, we obtain the lowest possible total cost reduction $\mathrm{o} = 1 - (1 - \mathrm{o_1}) \times (1 - \mathrm{o_2}) = 92.2\%$.

Thus, the minimum possible cost of writing b =71.25kb of data to the public decentralized Ethereum registry using two delegated transactions is only $\delta_{exp} \times (1 - o) = \$0.4$. Indeed, this is only a minimal estimate, which is highly dependent on the volatility of cryptocurrency prices and network load.[13] In the past, there were cases where a budget restriction of the decentralized system would not allow transactions to take place for several months or would have greatly exceeded that budget. In addition, the minimum value obtained does not consider the cost of any additional logic other than the plain write of the data obtained from experts and secure interaction through transaction delegation. Any additional information or logic, for example, when considering consistency of estimates,[14] will be charged additionally.

Conclusions

A universal method of transaction delegation has been developed and tested on the real network, which combines the advantages of existing methods, has a standardized backend server and client side, and at the same time does not require standardization of the decentralized application. This leads to a simplified interaction between user and DApp, as now user doesn't need to hold both DApp's and DDP's currencies in order to perform transactions. Suggested the use of the method for decentralization of expert evaluation systems within other systems, as well as some ways to optimize the cost of operation of such systems, which can result in a total cost reduction of 92.2 % compared to the typical, non-optimized cost of decentralized applications.

The developed method and a software for transaction delegation in decentralized data platforms facilitates the further adaptation of decentralized applications in all spheres of life and technology, since this method simplifies both the user experience and the necessary efforts for developers to implement the transaction delegation system in their projects.

The universal transaction delegation method was tested on a project that has nearly 2000 decentralized application's users (the number of users is approximated based on the number of addresses owning a project's token). This development is the open-source solution that can be adopted by any other project based on the decentralized Ethereum data platform.[15]

Acknowledgement

This research was supported by CyRADARS project (SPS G5286 “Cyber Rapid Analysis for Defense Awareness of Real-time Situation”) in the frame of the NATO Science for Peace and Security program.

References

1 Sunny King and Scott Nadal, "Ppcoin: Peer-to-peer crypto-currency with proof-of-stake," self-published paper (August 19 2012): 1.

2 Patrick Schueffel, "Alternative distributed ledger technologies Blockchain vs. Tangle vs. Hashgraph-A high-level overview and comparison," Tangle vs. Hashgraph-A High-Level Overview and Comparison (2017).

3 Mykyta Savchenko, Vitaliy Tsyganok and Oleh Andriichuk, “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.

4 Hung Dang, Tien Dinh, Eechien Chang, Qian Lin, and Beng Ooi, "Towards scaling blockchain systems via sharding," *Proceedings of the 2019 International Conference on Management of Data* (*SIGMOD '19),* 2019, pp. 123–140.

5 Dilip Mookherjee, “Decentralization, hierarchies, and incentives: A mechanism design perspective,” *Journal of Economic Literature* 44, no. 2 (2006): 367-390.

6 Siraj Raval, *Decentralized Applications: Harnessing Bitcoin's Blockchain Technology* (O'Reilly Media, Inc., 2016), 6-7.

7 Andrew Cullen, Pietro Ferraro, Christopher King, and Robert Shorten, "Distributed ledger technology for IoT: Parasite chain attacks," *arXiv preprint arXiv:1904.00996* (2019).

8 Dmitry Khovratovich and Jason Law, “BIP32-Ed25519: Hierarchical Deterministic Keys over a Non-linear Keyspace,” *2017 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW) Paris, France,* 2017, pp. 27-31.

9 Myrto Arapinis, Andriana Gkaniatsou, Dimitris Karakostas, and Aggelos Kiayias, “A Formal Treatment of Hardware Wallets,” *IACR Cryptology ePrint Archive* (2019): 34.

[10] M. Pustišek and A. Kos, "Approaches to front-end IoT application development for the ethereumblockchain," *Procedia Computer Science* 129 (2018): 410-419.

[11] Hossein Rezaeighaleh and Cliff C. Zou, "New Secure Approach to Backup Cryptocurrency Wallets," In *submission to IEEE Global Communications Conference-Communication & Information Systems Security Symposium*, *2019.*

[12] Mark Kim, Brian Hilton, Zach Burks, and Jordan Reyes, "Integrating Blockchain, Smart Contract-Tokens, and IoT to Design a Food Traceability Solution*," 2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), November* 2018, pp. 335-340.

[13] Krzysztof Wołk, "Advanced social media sentiment analysis for short-term cryptocurrency price prediction," *Expert Systems,* 2019, DOI: 10.1111/exsy.12493.

[14] Vitaliy Tsyganok and Sergii V. Kadenko, "On sufficiency of the consistency level of group ordinal estimates," *Journal of Automation and Information Sciences* 42, no. 8 (2010): 42-47.

[15] Nikita Savchenko, "Ethereum Delegated Transactions Service," *Crypatograph DApp github.com*, 2019, https://github.com/ZitRos/ethereum-delegated-tx-service [accessed June 8, 2020].

About the Authors

Nikita **Savchenko** holds B.Sc. (2016) and M.Sc. (2018) in computer science from the National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute." Currently, he is a PhD student in the Institute for Information Recording of National Academy of Sciences of Ukraine.

Vitaliy **Tsyganok** holds a DSc degree in system analysis and optimum solutions theory from the Institute for Information Recording of National Academy of Sciences of Ukraine. Currently, he is head of Laboratory for Decision Support Systems in the Institute for Information Recording of National Academy of Sciences of Ukraine and professor in the National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute."

Oleh **Andriichuk** holds a M.Sc. degree in applied mathematics from the National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute" and PhD degree in system analysis and optimum solutions theory from the Institute for Information Recording of National Academy of Sciences of Ukraine. Currently, he is senior researcher in the Institute for Information Recording of National Academy of Sciences of Ukraine and lecturer in the National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute."