Notes on Intuitionistic Fuzzy Sets
Print ISSN 1310–4926, Online ISSN 2367–8283
2025, Volume 31, Number 4, 610–628
DOI: 10.7546/nifs.2025.31.4.610-628

Analysis of the uncertainty of the service of requests at the attestation stage of a blockchain in an overall telecommunication system

Ivan Kavaldzhiev [1], **Velin Andonov** [2] **and Stoyan Poryazov** [3]

[1] Institute of Mathematics and Informatics, Bulgarian Academy of Sciences
Acad. G. Bonchev Str., bl. 8, Sofia-1113, Bulgaria
e-mail: i.kavaldzhiev@math.bas.bg

[2] Institute of Mathematics and Informatics, Bulgarian Academy of Sciences
Acad. G. Bonchev Str., bl. 8, Sofia-1113, Bulgaria
e-mail: velin_andonov@math.bas.bg

[3] Institute of Mathematics and Informatics, Bulgarian Academy of Sciences
Acad. G. Bonchev Str., bl. 8, Sofia-1113, Bulgaria
e-mail: stoyan@math.bas.bg

Received: 18 October 2025 **Revised:** 22 November 2025
Accepted: 27 November 2025 **Online First:** 18 December 2025

Abstract: An overall telecommunication system including a blockchain is considered. The inclusion of the blockchain is aimed at improving the security of servicing of the requests and the overall Quality of Service (QoS) of the system. Analysis and classification of the possible sources of uncertainty in block attestation and finalization is presented. Intuitionistic fuzzy estimation of the impact of the block finalization on the overall telecomunication system is obtained in the form of intuitionistic fuzzy pairs. An indicator for overall network efficiency is defined which takes into account the uncertainty related to the blockchain.
Keywords: Intuitionistic fuzzy set, Blockchain, Overall telecommunication system.
2020 Mathematics Subject Classification: 03E72, 68M10.

1 Introduction

Blockchain is an emerging technology that brings more and more utilization including fast payments between users all across the globe, secure cryptographically formed identity management systems, having immutable permanent ledger of history of transactions, making it suitable for supply chain management and other systems relying on such information that cannot be forged [11]. In addition, it can store non-fungable tokens of real life objects like living properties, certificates, art forms or objects part of the virtual world like game items. It has a self-regulated protocol that does not need a central mediator or authority, which brings higher trust than traditional centralized client-server based technologies. There are several different types of protocols that make it possible such blockchain based networks to be independent and self-organized. They are called consensus mechanisms. Some of the most popular ones are Proof-of-Work and Proof-of-Stake. Since most of the new blockchain networks transition more and more to Proof-of-Stake, which is more resource friendly, we will use it as a primary example.

The basic idea of Proof-of-Stake is that we have a special type of blockchain operators/nodes called validators, where each of them should stake a minimum amount of crypto tokens in the given blockchain network, in order to have the right to vote about which block should be the next one added to the chain [6]. The main incentive here is that if a validator is having malicious behaviour like deliberately postponing their vote, missing to vote or vote for an incorrect block — a slashing mechanism takes place, where the validator loses part or all of its stake amount. In this way all validators have the incentive to be honest, fair and keep the correctness of the chain of blocks.

In this article we will analyze several aspects of uncertainty which can happen during the process of block attestation. We will take Ethereum as one of the oldest and most popular blockchain networks, which transitioned from Proof-of-Work to Proof-of-Stake during the "Merge" in the Paris fork [9].

The Intuitionistic Fuzzy Sets (IFSs, see [2,3]) have proven to be an effective tool for modeling the uncertainty, vagueness and imprecision in complex problems accross various fields. More specifically, the concept of Intuitionistic Fuzzy Pair (IFP, see [4]), has been used for quantification of the uncertainty in the service of requests in overall telecommunication systems. In a series of papers (see [15–17]), an overall approach to the conceptual and analytical modeling of the uncertainty in service systems in general has been developed. Using this approach, we aim to obtain intuitionistic fuzzy evaluation of the uncertainty of the service of requests in the overall system which takes into account the uncertainty during block attestation.

2 Problem formulation

The analysis will be conducted using a specific use case for the blockchain network. It will be used to enhance a conceptual model of a telecommunication system serving users. Traditionally, a telecommunication network cannot guarantee the authenticity of the user making a call or request and the authenticity of the user who is the recipient of the call or request. Using special identifiers stored in the blockchain, a telecommunication system can be enriched to provide better security

by performing additional verification steps for each participant in the network, who chooses to benefit from this increased quality of service.

In Figure 1, we can see the conceptual model of the telecommunication system, which is utilizing a blockchain network to securely store the identities of the users in its immutable ledger. In the A-user verifying and B-user verifying stages we can see that the Telecom network is performing requests to the Blockchain to validate the users. These are the only points of communication between the two systems (Telecom network and Blockchain). In this research, we will inspect further what types of uncertainties might be expected during execution of requests to the blockchain and how this affects the performance and execution of the overall telecom system.

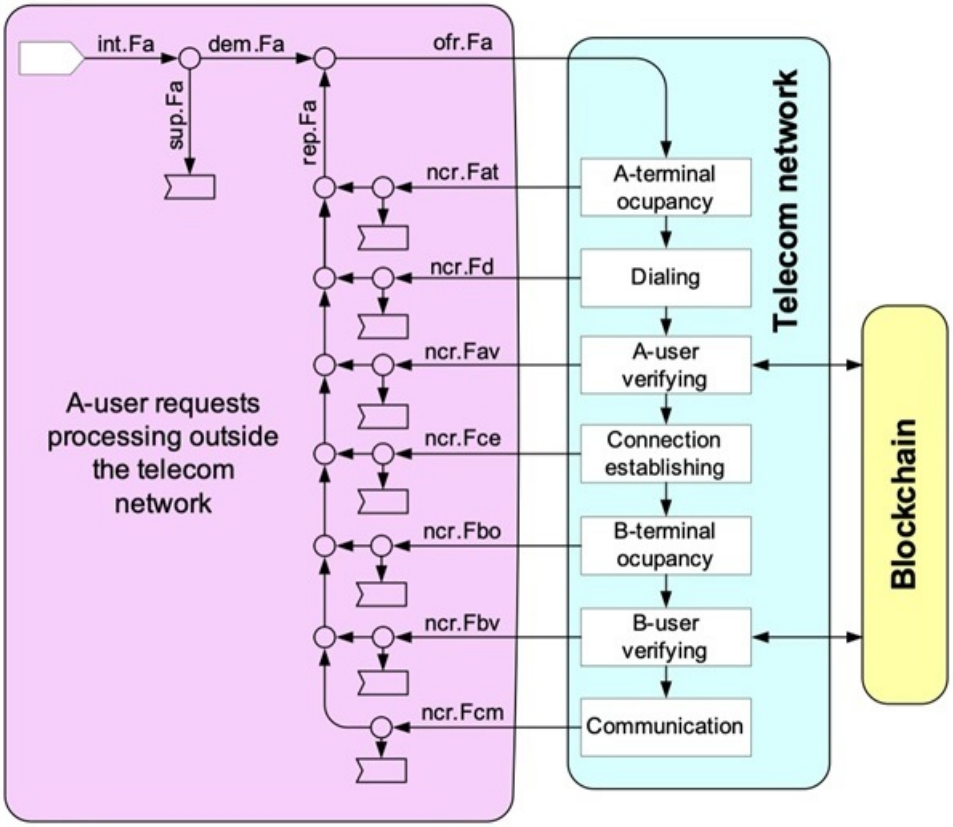

Figure 1. Conceptual model of an overall telecommunication system including a blockchain.

3 Consensus layer in Ethereum

Let us see how the consensus mechanism is constructed in Ethereum. The time in which the network operates is divided into epochs (around 6.4 minutes), where each epoch is split into 32 slots (each of them is in 12 seconds). This organizes the network to have strict rules based on the time and keeps it live and moving forward [7].

The consensus part of the network consists of peer-to-peer nodes called validators. In order to become a validator and participate in the block attestation you should stake 32 ETH (Ether) which is the native crypto currency of the network. The whole lifecycle of the block attestation has several types of validators, serving a different role:

1. Block proposer — a validator that is chosen to be a block proposer at a given slot of a specific epoch. That is the validator that bundles transactions into a block and gossips it to the network.
2. Block attester — a validator whose responsibility is to accept a proposed block from the network, to execute and validate it and based on its current state and local knowledge about the network to vote for a proposed new target endpoint in the chain.
3. Aggregator — a validator that collects all of the votes from all attesters for the slot in a private sub-net using BLS cryptography, in order to save space and prevent the network from handling too much signatures.

The consensus layer relies on the combination of several algorithms. These are the RANDAO, LMD GHOST and Casper FFG [7]. All of the participating validators are part of a separate subchain called beacon chain, which is responsible for keeping new blocks getting proposed, collecting votes for them and eventually reaching finality for the ones that accumulated the most staking power.

The RANDAO mechanism is the one responsible for assigning duties for all of the validators. It determines what role each validator will have for a given epoch [7]. This prevents attackers from knowing too much in advance what role a validator will perform and from organizing specific attacks based on the duties of all validators. In addition, this algorithm provides fairness to the blockchain, since validators constantly change their roles and duties in a random manner, thus having as much right to take every type of responsibility in the beacon chain, as possible.

The GHOST algorithm stands for "Greedy Heaviest-Observed Sub-Tree". In contrast with the Bitcoin fork choice rule, where the longest chain among all chains in a given fork is chosen to be the canonical one, the GHOST implies that the subtree starting from a given block, chosen to be the head of the blockchain, has the largest amount of stake power. Voting for such a block to be the head of the blockchain means that a validator votes for all ancestors of this block. The LMD stands for "latest message driven", meaning that the liveliness of the whole network is driven not by block producing but by block voting/attesting. Not only that, but the consensus layer takes into account the latest vote sent by a given attester. All previous votes from a validator are discarded. Each participant of the network is keeping its own local copy of the network view, that is built by the blocks and attestations that were consumed. The LMD GHOST represents a fork choice rule, where the node chooses the best head of the blockchain based on this local view. The choice might differ from the choices of the other node peers, but that is considered normal. Each node is voting in the best possible way based on the local information they have. Eventually, the whole network should consolidate to the same block as the new head of the block tree [7].

Casper FFG serves the role of a finality gadget that determines finality over the LMD GHOST algorithm. It utilizes the fact that we have a given predefined set of validators that hold all of the ether stake amount. Then, following the consensus theory and the Byzantine fault tolerant (BFT) protocol, at maximum $1/3$ of all validators can be faulty or give wrong votes in order for the algorithm to advance in the correct way. FFG stands for "Friendly Finality Gadget". As the name denotes, it is a utility mechanism that adds on top of the consensus layer. It serves to determine finality of a given block or basically a new target that is the newest current checkpoint of the

blockchain. All other ancestors of this block or new checkpoint are also considered valid and immutable. This algorithm guarantees that the network advances by having constantly updating finalized checkpoints and the history kept inside the blockchain can be trusted [7].

Let us see how all of these mechanisms come into place. In Ethereum, we have a very large number of validators that are participants. It would be very impractical and possibly impossible to gossip and count the votes of several hundred thousands of validators, especially having in mind the network will scale with more and more users over time. That's why voting is spread in equal periods of time called epochs. Each epoch is further divided into 32 slots, where one slot is 12 seconds. All of the validators will cast their votes in a given epoch, which means in each slot $1/32$ of all validators will be determined to participate.

Similarly, in order for the consensus layer to have better performance and save storage, validators picked to perform attestation for a slot are grouped in committees [10]. Each committee has at least 128 validators. One of these validators is the aggregator, who has the special role to combine all of the signatures from the validators in the committee that have the same attestation [10].

In order for the votes to be unified and more easily determined, all validators are attesting for a new checkpoint in an epoch. This is the first slot of a given epoch and usually each slot is connected to a new block [7]. Here is a breakdown of an attestation from a given validator and its contents:

- aggregation_bits — a bitlist of validators, where we have a mapping between the position of a validator in a committee and a value 0 or 1 showing whether the validator has signed the data;
- data — the specific details related to what the validator is attesting for;
- signature — a BLS signature aggregating the signatures of the individual validators [10]. These signatures are aggregated only from the validators performing the same attestation.

When a validator is an attester for a given slot, they should construct the needed data (listed above). It consists of the following fields:

- slot — the slot number the attestation is made to;
- index — a number identifying which committee the validator is part of for the specified slot;
- beacon_block_root — the result of the fork-choice algorithm - the root hash of the block the validator thinks is the head of the chain;
- source — pointing to what the validators see as the most recent justified block;
- target — pointing to what the validators see as the first block in the current epoch [10].

When the data is created, the validator flips their corresponding bit in the aggregation_bits to show their participation. Then, the validator signs the attestation and gossips it to the beacon chain [10].

The Casper FFG algorithm helps for finalizing these checkpoints, so it finalizes the first slot of a given epoch. All slots from all previous epochs are also finalized, but the current epoch N has 31 slots which are not yet finalized. It works in a 2-phase process in the way described below.

In the first voting cycle a node X makes a judgement based on its local view of the current epoch's checkpoint. This information is broadcasted to the network and in parallel the node X is awaiting the views of the network from the rest of the participants. If more than $2/3$ of the nodes tell to node X that they also share the same view for the checkpoint, then node X marks the checkpoint as justified. In the second cycle, node X notifies the rest of the network that it knows a supermajority of the network has agreed that the checkpoint is valid one or in other words — node X notifies that it has marked the checkpoint as justified. The node also hears from the network whether the rest of the validators know that a supermajority of the nodes think that the checkpoint is valid. If so, node X marks the checkpoint as finalized. Once finalized, no honest nodes will ever revert it or they will be slashed. A key note here is that only attestations that are recorded inside blocks are taken into account and votes are gossiped to the network also by adding them into a block.

The anatomy of a vote using the Casper FFG algorithm consists of a link $s \rightarrow t$ between a source checkpoint and target. Choosing a target t as a final for the network, means that a supermajority of the validators have voted for the same link $s \rightarrow t$. The supermajority is in the context of having votes that are backed by a supermajority of the staked value inside the network, not on the voting counts themselves [7].

Slashing is the mechanism used to guarantee safety and proper behaviour of all participants, which are validators. If there is a malicious actor all or part of their stake will be removed and the node will be removed from the set of validators. This makes it very unlikely a validator to be acting improperly. Opposed to this, honest validators have an incentive to give correct votes for the link by receiving different types of rewards — a source vote reward and a target vote reward.

4 Uncertainty in block attestation and finalization

While the Proof-of-Stake algorithm utilizing the Gasper protocol is very powerful and guarantees the network liveliness and safety there are a couple of corner cases or ways in which the block attestation process can be skewed in such a way that it can disrupt the liveliness of the network and delay the block finalization with a long period of time [5].

One point of uncertainty in the PoS (Proof-of-Stake) lifecycle is the moment of reaching finality for a block. This can happen during several types of attacks or infrastructure issues. Since Ethereum is a permissionless network, everyone with 32 of staked Ether can install the needed software client and join the network as a validator. What is more, an attacker with a higher budget can have ownership of multiple validators or have a huge amount of staked Ether, increasing their voting power. We will take a look how such actors as well as infrastructure problems that might appear can influence the amount of certainty for executing a request in the network.

4.1 Reorg attacks

These types of attacks are related to replacing blocks on the canonical chain. This can be done even with a small amount of staked Eth. There are already existing attacks by having a malicious validator attesting for a block B for a slot $n + 1$, but keeping the vote in private until slot $n + 2$. Meanwhile, block C appears and an honest validator proposes block C for slot $n + 2$. Near the

same time, the attacker releases all attestations for block B and it manages to get more attestations than block C. Then when the next block D appears it has more staking weight when it's added on top of block B and thus, block C is effectively discarded, even though it was honestly generated and proposed [8]. That's an example for a 1 slot ex-ante reorg. It can be easily executed by having 34% of the stake but can be attempted even having a lower stake than that. This type of attack creates uncertainty of rejecting valid blocks during block finalization. This happens at the stage of majority voting for target $T + 1$ (see Figure 2).

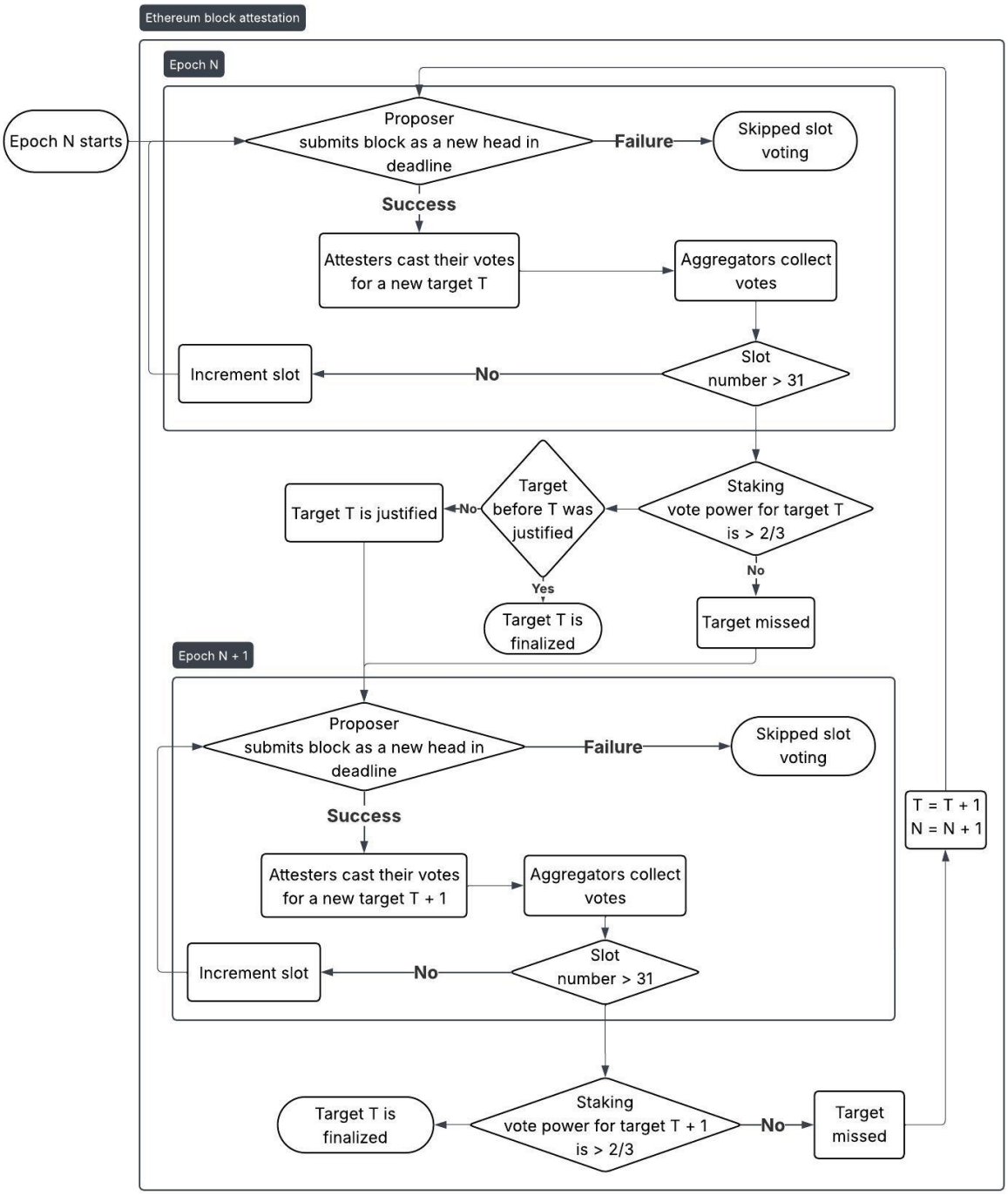

Figure 2. A flowchart of the service of requests at the Consensus layer in Ethereum.

4.2 Balancing attack

A more comprehensive way to reorg is the balancing attack. It uses the LMD (Latest Message Driven) concept where validators take as a valid only the latest attestation that they have received from a given node. Each validator keeps a table of the "latest message" received from each of the rest of the validators [13]. Then when a validator receives a message with attestation from another validator it replaces its latest message entry if and only if the new incoming message has a slot higher than the one present in the table. This means that if a validator receives two messages for the same slot from the same validator, only the earlier one will be kept and stored. It is possible for an attacker to release such attestation messages to the network that can make it possible half of the network to receive the first message first and the other half to receive the second message first. That's a key point to making the attack possible.

The attack contains two steps:

- Adversarial block proposers initiate two competing chains — Left and Right by sending votes for different blocks at different parts of the network. This is against the fork-choice rule and the validator will be slashed, however these efforts will initiate the beginning of the attack, continued by other malicious validators [8].
- From time to time a certain number of votes per slot from the attacker proposers are needed to keep the whole network in a tie by having half of the honest validators believing the Left one is the correct one and the other half believing that the Right one is correct, creating a tie in the network [5, 12, 13].

After some time all honest validators will receive enough attestation to conclude that there is a split view of the network, but for some time, attacker nodes can trick the network to keep believing on their half of the view and continue maintaining it. This can cause uncertainty for block finalization with an undefined amount of time about a valid block being justified and eventually finalized. This happens at the stage of majority voting for target $T + 1$ (see Figure 2).

A way to get out of this attack is proposer boosting [7] or utilizing the Ethereum-TrInc approach described in [5]. The proposer boosting allows when a proper and valid block is consumed by a validator, temporarily a huge amount of staking power to be added to the branch formed by this new block during fork choice calculations. The exact value is determined by the PROPOSER_SCORE_BOOST property, which determines the staking percent power of all of the validators assigned to vote in the same slot.

4.3 Bouncing attack

This attack is also known as flip-flopping. Here the attacking validators aim to achieve justification of checkpoints that constantly change between fork A and fork B. That can be only achieved by having a very fine control over the network latency [8]. The constant switch of justification points between these two forks, make it so we don't have matching pairs of justified source and target checkpoints that can achieve finality. This disrupts the finality mechanism of the network.

The network has a defence mechanism which is proposer-weight boosting — favouring attestations received in a timely manner (usually the first 4 seconds of a new slot). In addition

the fork-choice rule was enhanced, so that the network can choose to switch to another block fork only in the first $1/3$ number of slots of a given epoch. Later attempts to do so, will not work. This mechanism counterattacks the malicious validators which are withholding blocks and attestations for later gossipping. The proposer-weight boosting, however, has the most strength against validators that have a small amount of stake in the network. Attackers with big stake might get around this defense. This attack creates uncertainty of block finalization by the amount of time needed for the proposer-weight boosting to take place. In Figure 2 this happens at the stages of majority voting for target T and $T + 1$.

4.4 Finality delay

Another type of an attack is where a given malicious attacker has control of over 30% of the staking amount in the network and wants to delay finality by performing a length-n-finality delay [14]. This can happen when they are the block proposer for the epoch boundary block — let's say block 32 for epoch X and the next block 33. They create a private fork and wait until there are enough attestations for a previous epoch block that is the target epoch boundary block, e.g. block 31. That will happen since the network is not aware of the existence of block 32 yet. Then the attacker reveals the private fork and block 32 starts to be recognized as the target block for epoch 1. However, votes for block 31 are then treated as incorrect and block 32 cannot collect enough attestations, thus no block is justified [14]. This attack is indeed a finality delay one, since after a future epoch is finalized, all epochs before it will also be finalized. The RANDAO mechanism should guarantee that an attacker cannot always be in the same suitable position of proposing two consequent blocks in the boundary of an epoch.

The solution for such an attack is entering a special state of the network called "inactivity leak" mode. That happens if the network has not finalized a checkpoint for a number of epochs that is defined by a parameter called MIN_EPOCHS_TO_INACTIVITY_PENALTY with default value of 4 [7]. After entering in such a state the following rules related to rewards and penalties apply:

- attestation penalties are kept, but attestation rewards are removed;
- if a validator is registered to be inactive, their inactive score is raised. This score determines the inactivity penalty that can raise quadratically. This is how the inactivity leak is realised;
- rewards for proposer and sync committee are kept the same.

When the minimum number of epochs with no finalized checkpoints is reached, the network is gradually leaking the stake of inactive validators by basically burning part of their stake amount. After some time, the remaining active validators will get back the control of more than $2/3$ of the stake in the network. This will make it possible to have finalized checkpoints again in the network [7]. Here we have uncertainty for block finalization for an undefined period of time needed to get the network having state power which is at least 67% of the overall staked amount. In Figure 2 this happens at the stage of majority voting for target T in Epoch N.

4.5 Double finality

This type of attack can happen if the malicious actor has 34% of the overall stake of the network [8]. The way the attack is done is in the following steps:

- when the attacker validator is block proposer, they equivocate by creating two different forks;
- then vote for both of these with their validator/validators.

This creates two forks which have 34% of staking vote power used for both of them. What they need is 50% of the remaining 66% stake voting power ($34\% + 33\% = 67\% > 2/3$ of the stake) to vote for them and they both will be able to be finalized, causing a double finality. In other words, the malicious node should have a very fine tuned control over the timing in the network, so that an equal amount of the rest of the validators receive each of the forks, so that they vote for them. Only then, the attack will be successful and the network will have two separate finalized forks. If such an event happens the solution is to use the so called "social consensus" where honest validators and the overall community decide which of the forks to choose for canonical one.

As a result of this attack, the malicious validator will destroy all of the 34% stake they own, since the validators will violate the fork-choice rules by conducting double voting. We have uncertainty whether the social consensus will choose as a canonical, the chain containing the block of our interest or if they choose it, what time it will take. In Figure 2, we can imagine the same steps and process happen twice in different halves of the blockchain network and the attack is connected to having a majority voting for target $T + 1$.

4.6 Attacks related to having greater or equal to 50% of the staked amount

If a dishonest validator has exact control of 50% of the stake power, they will be able to create a second separate malicious fork and by also voting to the honestly selected fork, they will maintain the two forks. This will break finality and an inactivity leak procedure will be triggered, causing also loss to a lot of the staking power of the malicious actor. Additionally, it will be very unlikely for a bad actor to have exactly 50% of the stake, which is a dynamically changing value [8]. We have uncertainty of block finalization described by the undefined time of an inactivity leak getting triggered. This happens at the stage of majority voting for target $T + 1$ (see Figure 2).

If the attacker has greater than 50% they will be able to dominate (note — not fully control) the attestation process combining honest nodes to agree and keep a maliciously built fork view of the network. Then some reorgs might happen, some blocks might be censored. This will create uncertainty of block attestation service and lead to denial of handling certain blocks. If such events occur, then again social consensus will take place and honest nodes should pick an honest minority fork as the new canonical chain [8].

And then if the attacker has greater than 66% voting power, which would mean a bigger stake than $2/3$, it can have full control over the fork choice rule and selects whatever blocks they want in the canonical chain. This would mean they can spend some value, then revert the transfer,

then back spend it and perform double spend attacks. All of these malicious activities could be finalized and baked into the network [8]. Even if honest nodes had the intention to attest for the canonical chain, they will get slashed for voting for a chain with less staking weight.

Again the social consensus, also called Layer 0, should take place. The whole community should decide for a resolution of the corrupted chain. They can decide to build on top of the maliciously created chain or start fresh from a saved checkpoint before the attack, even though this fork would have less staking weight.

For the cases where the attacker has control of over 50% of the stake, we have uncertainty of block finalization described by the undefined time of a social consensus taking place. This happens at the stage of majority voting for target $T + 1$ (see Figure 2).

4.7 Other attacks

There are other types of attacks which are possible in a Proof-of-Stake based blockchains, however Ethereum gets around them by using Casper and the LMD GHOST algorithms [10]. Examples for such attacks are avalanche attacks, where an attacker proposes a privately built block subtree and creates equivocation votes for different blocks. This would create different forks and delay finality. Due to the LMD component of the GHOST algorithm, validators count only the first received vote for a given slot by a particular attester. This removes the application for this attack. Another potential risk is the nothing-at-stake problem. If a validator is a participant in the network without staking anything, they can vote for multiple forks or no fork at all, without any penalties. Ethereum secures its consensus layer by having a slashing mechanism to punish incorrect behaviour. There are also other types of attacks, which are not relevant for the security of Ethereum and will not affect the analysis of uncertainty in request serving. That is why we will not cover them in this research.

4.8 Infrastructural issues

Malicious attacks to the network are not the only cases where we can observe uncertainty in execution. Different portions of the network validators can experience network issues. This means that all of their stake will not be available and cannot be used to affect the voting of the block. If the majority of stake is not available due to this issue, this means no finality will be reached. Then the inactivity leak mechanism will apply, so the uncertainty of execution is valid based on the value of the MIN_EPOCHS_TO_INACTIVITY_PENALTY property. This happens at the stage of majority voting for target $T + 1$ (see Figure 2).

Very high latency or drops of network packages can cause delayed attestation for some validators. This means they might miss voting for the canonical chain and miss the 4 seconds rule of sending attestations. This can cause delay of reaching justification or finality and if the network latency is not repetitive and constant and happens to different validators, then even if inactivity leak triggers due to node inactivity, there is an undefined uncertainty of when exactly block finalization will become stable. In Figure 2 this happens at the stages of majority voting for targets T and $T + 1$.

Network delay and latency can also cause validators not seeing fast enough the block subtree with the biggest staking weight and having skewed local view of the whole network. This might cause voting for different subtrees with less staking weight, creating separate forks maintained for some time. The issue remains until the network latency stabilizes and connections improve. Then validators who experienced poor connection will catch up and start voting for the canonical chain. The problem will resolve by itself due to the LMD GHOST algorithm. This also creates uncertainty of block handling with an undefined period of the moment network connection gets stable for the majority of participants. This happens at the stage of majority voting for target $T+1$ (see Figure 2).

Poor connectivity experienced by a high number of validators, can also lead to entirely skipped votes and blocks gossiped at different rates, all of which can lead to temporary reorgs of blocks. This means a valid block part of the canonical chain being overridden by a block coming from a subtree of less attester weight. This leads to uncertainty of block execution by experiencing temporary instability of the network by wrong block organization. This creates uncertainty of rejecting valid blocks due to reorganizations. This happens at the stage of majority voting for target $T + 1$ (see Figure 2).

5 Uncertainty impact of block finalization on an overall telecommunication system model

In Figure 1, we can see a conceptual model of a telecom system using blockchain. Blockchain will increase the security of service of the network and overall the quality of service. The uncertainty elements we inspected in the previous chapters are happening entirely in the Blockchain component (far-most right) on the diagram. The block scheme in Figure 2 shows specifically the consensus part of this Blockchain component. Having uncertainty in the block finalization means that if an executed transaction is not part of a finalized block, its results are not yet to be trusted and there is no guarantee the information read or written from state is currently correct or is ever going to be. Once finalization is reached, we have a full guarantee that the transaction result reflects permanent changes on the state. The only way to break this guarantee is having a malicious validator with greater than $2/3$ of the overall stake power (greater than 66%) that can overwrite block history the way they want. However, this will lead to enormous loss in terms of slashed state amounts resulting in millions of Ethers.

As already mentioned, Figure 2 describes entirely the consensus layer of the Ethereum blockchain lifecycle. There is a preceding logic covering the interception of user requests in the form of transactions. When a user submits a transaction, eventually it will get into a block. This will be done via a block proposer, who selects the transaction for inclusion in a newly proposed block. This means that in Figure 2 we cover the initial moment of submitting the block containing this transaction in Epoch N. Then, when a block or the new target T is finalized, the lifecycle for a given block ends and the user can poll the blockchain to inspect when the block is finalized and be sure that their transaction will remain secure and immutable.

In order to quantify the uncertainty in the service of requests, we shall apply the approach

described in [16, 17]. At the base of this approach stays the notion of a base virtual service device. Each base virtual device x has the following parameters: intensity of the flow of requests (Fx), probability of directing the flow of requests towards the device (Px), service time in the device (Tx), traffic intensity (Yx, measured in Erlang).

To enhance the construction of conceptual and analytical models of service systems in general, special notation of the parameters of the virtual service devices is proposed. It uses qualifiers which characterize the traffic. The names of the devices are in small or subscript letters. For example, $scc.Fx$ is the intensity of the carried flow of requests of the device x. In the figures, only the names of causal devices may be present. The names of the device parameters are implicit.

In [17], three ways to characterize the uncertainty in the service of requests inside a virtual service device are discussed: IF traffic, IF flow and IF probability estimation. They are based on an IF characterization of the traffic inside a virtual service device (see Figure 3).

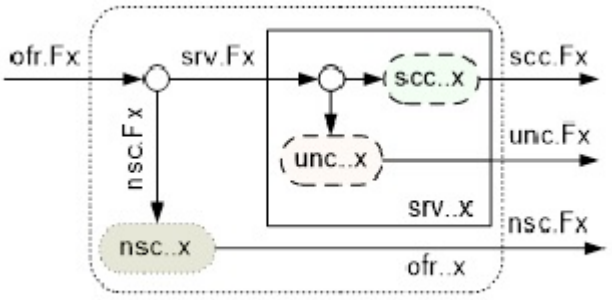

Figure 3. Causal decomposition of the traffic inside a virtual service device x (see [17])

.

For the definitions of not served, offered, served, uncertain and successful traffic, see [17]. The qualifiers may be two, one or none. In case that the parameter's symbol is omitted, the causal name is a name of a device (see Figure 3).

Now, the three intuitionistic fuzzy characterizations of the service of the requests by device x are defined as follows.

1. IF traffic characterization:

$$\mu_x^y = \frac{scc.Yx}{ofr.Yx}; \nu_x^y = \frac{nsc.Yx}{ofr.Yx}; \pi_x^y = \frac{unc.Yx}{ofr.Yx}\,. \tag{1}$$

2. IF flow characterization:

$$\mu_x^f = \frac{scc.Fx}{ofr.Fx}; \nu_x^f = \frac{nsc.Fx}{ofr.Fx}; \pi_x^f = \frac{unc.Fx}{ofr.Fx}\,. \tag{2}$$

3. The IF time characterization, considered in [15] is equivalent to the IF traffic characterization.

In Figure 4, a conceptual model of the uncertainty immpact of block finalization on the overall telecommunication system is shown.It describes the flow of requests, which are incoming. Part of them enter the blockchain network in the form of transactions. Then when these transactions are selected to be part of a newly proposed block, we can see the block processing logic in Figure 2.

Under conditions which make a block not valid or it enters the situation of reaching uncertainty in attestation, in Figure 4 we can see that the user either decides to terminate the execution or make a repeated attempt. When in Figure 2 a block is successfully finalized, then in Figure 4 we see how the request from the user related to the blockchain is successfully executed and the processing of the telecom request proceeds to be handled by the next network stage.

Since the telecommunication network has two communication stages with the Blockchain, which are A-user verifying and B-user verifying, the uncertainty of block finalization has an impact on these two virtual devices only. In Figure 4 we can see in more detail how exactly incoming traffic from the verification stages of the telecomunication system is served inside the blockchain. It describes the A-user verification stage, but the exact same logic will apply also to the B-user verification stage.

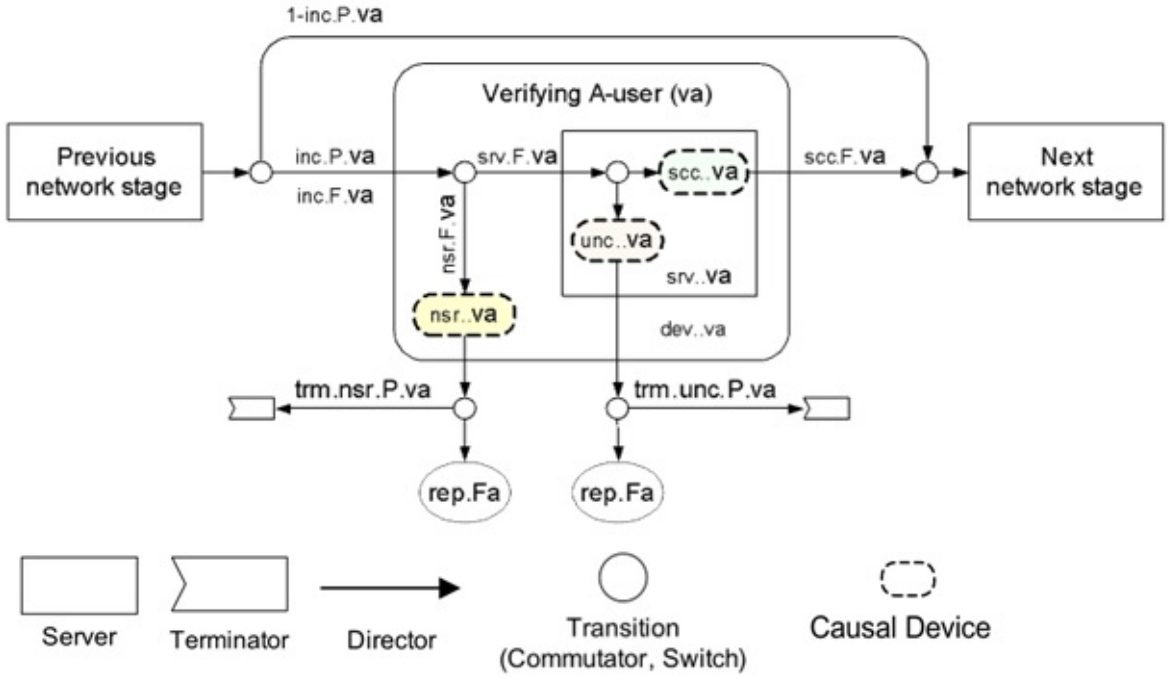

Figure 4. Uncertainty impact of block finalization on an overall telecommunication system.

First, the incoming flow of requests is split into two categories. With a probability of $inc.P.va$ (qualifier inc. comes from "incoming") the traffic is directed to blockchain for verification purposes. And with a probability of $1 - inc.P.va$ the traffic bypasses the blockchain. In this case, the users might not want to pay additionally for the improved security and will experience zero delay and lack of uncertainty by going directly to the next network stage.

Looking at the verification part in more details, we can see that the users can experience one of two main categories of request processing:

1. $nsr.va$ — not served traffic. There is a request which has a malformed structure, incorrect or missing structure and does not enter the gossipping phase of the blockchain at all. The blockchain node that has first received the request will directly reject it. Then the request is either terminated with probability $trm.nsr.P.va$ as not served or the user can choose to repeat the execution by adding it to the repeated flow of $rep.Fa$.

2. $srv.va$ — served traffic. The request reached execution of the blockchain and was propagated inside the whole network. Here the traffic has two subcategories:

   - $scc.va$ — the request is successfully executed with a clear status — the blockchain has returned a concrete result for the request — either positive or negative. Then the execution continues with the next network stage.
   - $unc.va$ — the request is served with uncertainty. This is the part where all of the discussed issues related to block finalization have their effect. If the blockchain experiences uncertainty in the phase of block attestation and/or finalization, the request will remain in uncertain status. At this stage, the user can decide whether to terminate it with probability $trm.unc.P.va$ or add it to the repeated flow of $rep.Fa$ and execute the request again.

These sub-stages define how the blockchain affects the overall efficiency of service of the telecommunication network. It can be measured by using μ coefficient denoting the quality of service for a given stage. Looking back at Figure 1 we can see seven stages of execution between user A and the telecommunication network. Each of these stages can be defined with a μ factor, so overall we have coefficients from μ_1 to μ_7. The stages that communicate with the blockchain are the ones linked to μ_3 and μ_6.

We may consider the service in the third stage (verifying A user, in Figure 4), as composition of two parallel services: verification (with probability $inc.P.va$) and bypass verification (with probability $1 - inc.P.va$). In [17] the intuitionistic fuzzy estimation (IFE) of a composition of two parallel services is derived. In general, there are four IFEs corresponding to the teletraffic parameters. These are:

1. Probability of directing the served requests to the service device.
2. Incoming requests' flow intensity.
3. Service time duration of a request.
4. Served traffic intensity.

The values of the time and traffic IFEs coincide. The values of the probability and flow IFEs coincide. In our case, (see Figure 4), for simplicity, we consider the probability IFE. The traffic IFE can be easily derived analogously, using the method described in [17].

The service composition in stage 3 has an IFE in the form of an intuitionistic fuzzy pair (IFP, see [4]): $\langle \mu_3, \nu_3 \rangle$ and degree of uncertainty π_3. The composed service functions are verification (with IFE $\langle \mu_{3v}, \nu_{3v} \rangle$) and bypass (with IFE $\langle \mu_{3b}, \nu_{3b} \rangle$). Following [17] and Figure 4, we obtain the below expressions for the IFEs:

$$\mu_3 = (1 - inc.P.va)\mu_{3b} + inc.P.va\,\mu_{3v}\,, \tag{3}$$

$$\nu_3 = (1 - inc.P.va)\nu_{3b} + inc.P.va\,\nu_{3v}\,, \tag{4}$$

$$\pi_3 = (1 - inc.P.va)\pi_{3b} + inc.P.va\,\pi_{3v}\,. \tag{5}$$

The values of the verification IFE are results of the blockchain work in verifying A-user presented generally as performance of device $dev..va$ (Figure 4):

$$\mu_{3v} = (1 - nsr.P.va)(1 - unc.P.va)\,, \tag{6}$$

$$\nu_{3v} = nsr.P.va\,, \tag{7}$$

$$\pi_{3v} = (1 - nsr.P.va)unc.P.va = 1 - \mu_{3v} - \nu_{3v}\,. \tag{8}$$

Therefore the pair $\langle \mu_{3v}, \nu_{3v} \rangle$ is an IFE of the verifying service of A-user in stage 3.

The values of the bypass IFE are by assumption: $\mu_{3b} = 1$, $\nu_{3b} = 0$ and $\pi_{3b} = 0$. Using the values of the bypass IFE, (3), (4), (5), (6), (7) and (8), we obtain:

$$\mu_3 = inc.P.va[(1 - nsr.P.va)(1 - unc.P.va) - 1] + 1\,, \tag{9}$$

$$\nu_3 = inc.P.va\, nsr.P.va\,, \tag{10}$$

$$\pi_3 = inc.P.va(1 - nsr.P.va)unc.P.va = 1 - \mu_3 + \nu_3\,. \tag{11}$$

Therefore the pair $\langle \mu_3, \nu_3 \rangle$ is an IFE of the verifying A user stage (stage No.3).

Let us define an overall network efficiency indicator of handling the traffic in a telecommunication system enhanced with blockchain μ_N as the multiplication of the degrees of membership of the IFEs of all stages of the network, i.e.:

$$\mu_N = \mu_1\mu_2\mu_3\mu_4\mu_5\mu_6\mu_7\,, \tag{12}$$

where

- μ_1 is the degree of membership of the IFE of the A-terminal occupancy stage (see Figure 1) and it is obtained using (2);
- μ_2 is the degree of membership of the IFE of the Dialing stage;
- μ_3 is the degree of membership of the IFE of the A-user verifying stage (see eq. (9));
- μ_4 is the degree of membership of the IFE of the Connection establishing stage;
- μ_5 is the degree of membership of the IFE of the B-terminal occupancy stage;
- μ_6 is the degree of membership of the IFE of the B-user verifying stage and it is obtained from (9) by substituting the index 3 with 6 and va with vb;
- μ_7 is the degree of membership of the IFE of the Communication stage.

The analytical expression for the degree of non-membership of the IFE of the network considered as a serial composition of 7 devices is very complicated (see [1]). An estimation for this degree can be obtained by taking the maximum of the degrees of non-membership of the IFE of all 7 stages, i.e.:

$$\nu_N = \max\{\nu_i : 1 \leq i \leq 7\}\,, \tag{13}$$

where

- ν_1 is the degree of non-membership of the IFE of the A-terminal occupancy stage (see Figure 1) and it is obtained using (2);
- ν_2 is the degree of non-membership of the IFE of the Dialing stage;

- ν_3 is the degree of non-membership of the IFE of the A-user verifying stage (see eq. (10));
- ν_4 is the degree of non-membership of the IFE of the Connection establishing stage;
- ν_5 is the degree of non-membership of the IFE of the B-terminal occupancy stage;
- ν_6 is the degree of non-membership of the IFE of the B-user verifying stage and it is obtained from (10) by substituting the index 3 with 6 and va with vb;
- ν_7 is the degree of non-membership of the IFE of the Communication stage.

Obviously, $0 \leq \mu_N + \nu_N \leq 1$ and $\langle \mu_N, \nu_N \rangle$ is an IFP. It can be considered an IF network efficiency indicator of handling the traffic in a telecommunication system enhanced with a blockchain.

6 Conclusion

In this paper, we analyzed the complexity that happens during block finalization in Ethereum and in which points and circumstances we can expect uncertainty at this stage. This has an impact over any integration of a system with a blockchain network. In our case study, we consider an overall telecommunication system, which utilizes blockchain to increase security by validating users' authenticity. The uncertainty of block finalization in this case will have an impact over the validation of the users and can influence the security and the efficiency of the telecommunication network.

An approach to building conceptual and corresponding analytical models of intuitionistic fuzzy performance estimation of an overall telecommunication system which utilizes blockchain technology to increase security by validating users' authenticity are proposed.

Acknowledgements

The work of Ivan Kavaldzhiev is supported by the research project BG05SFPR001-3.004-0012 "Developing Innovative MEthods, Technologies and mOdels to support a Data driven Economy", funded under the procedure "Support for the development of project-based doctoral studies" from the Programme "Education 2021-2027", co-financed by the European Union.

The work of Velin Andonov and Stoyan Poryazov is supported by the project "Perspective Methods for Quality Prediction in the Next Generation Smart Informational Service Networks" (KP-06-N52/2) financed by the Bulgarian National Science Fund.

References

[1] Andonov, V., Poryazov, S., & Saranova, E. (2024). QoS characterization of some service compositions based on intuitionistic fuzzy pairs. *Notes on Intuitionistic Fuzzy Sets*, 30(2), 190–202.

[2] Atanassov, K. (1986). Intuitionistic fuzzy sets. *Fuzzy Sets and Systems*, 20(1), 87–96.

[3] Atanassov, K. (2012). *On Intuitionistic Fuzzy Sets Theory*, Springer, Berlin.

[4] Atanassov, K., Szmidt, E., & Kacprzyk, J. (2013). On intuitionistic fuzzy pairs. *Notes on Intuitionistic Fuzzy Sets*, 19(3), 1–13.

[5] Atwi, A., Guenou, Y.A., Potop-Butucaru, M., & Zaghdoudi, B. (2025). Mitigating Balancing Attack on Ethereum PoS. In: Barolli, L. (eds) Advanced Information Networking and Applications. AINA 2025. *Lecture Notes on Data Engineering and Communications Technologies*, Vol. 246. Springer, Cham.

[6] Buterin, V., & Griffith, V. (2017). Casper the Friendly Finality Gadget. arXiv:1710.09437.

[7] Edgington, B. (2023). Upgrading Ethereum. A technical handbook on Ethereum's move to proof of stake and beyond. Edition 0.3. Accessed on: Oct. 1, 2025. Available online at: `https://eth2book.info/capella/`.

[8] Ethereum Docs. (2025). *Ethereum proof-of-stake attack and defense*. Accessed on: Oct. 2, 2025. Available online at: `https://ethereum.org/bg/developers/docs/consensus-mechanisms/pos/attack-and-defense/`

[9] Ethereum Docs. (2025). *Timeline of all Ethereum forks (2014 to present).* Available online at: `https://ethereum.org/bg/ethereum-forks/`.

[10] Ethereum Docs. (2025). *Attestation*. Accessed on: Oct. 2, 2025. Available online at: `https://ethereum.org/bg/developers/docs/consensus-mechanisms/pos/`.

[11] Habib, G., Sharma, S., Ibrahim, S., Ahmad, I., Qureshi, S., & Ishfaq, M. (2022). Blockchain Technology: Benefits, Challenges, Applications, and Integration of Blockchain Technology with Cloud Computing. *Future Internet*, 14(11), Article ID 341.

[12] Natoli, Ch.,& Gramoli, V. (2017). The balance attack or why forkable blockchains are ill-suited for consortium. *47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, Denver, CO, USA, 579–590.

[13] Neu, J., Tas, E.N., & Tse, D. (2022). Two attacks on Proof-of-Stake GHOST/Ethereum. *Cryptology ePrint Archive*. Available online at: `https://eprint.iacr.org/2022/289.pdf`.

[14] Neuder, M., Moroz, D.J., Rao, R., & Parkes, D. C. (2021). Low-cost attacks on Ethereum 2.0 by sub-$1/3$ stakeholders. arXiv:2102.02247.

[15] Poryazov, S., Andonov, V., & Saranova, E. (2022). Three intuitionistic fuzzy estimations of uncertainty in service compositions. In: *Uncertainty and Imprecision in Decision Making and Decision Support: New Advances, Challenges, and Perspectives*; IWIFSGN BOS/SOR 2020. Lecture Notes in Networks and Systems; Springer: Cham, Switzerland, Volume 338, pp. 72–84.

[16] Poryazov, S., Andonov, V., Saranova, E., & Atanassov, K. (2022). Two approaches to the traffic quality intuitionistic fuzzy estimation of service compositions. *Mathematics*, 2022, 10, Article ID 4439.

[17] Poryazov, S., Andonov, V., & Saranova, E. (2022). Intuitionistic fuzzy estimations of uncertainty of a parallel composition of services. In: *Intelligent and Fuzzy Systems*; INFUS 2022. Lecture Notes in Networks and Systems; Kahraman, C., Tolga, A.C., Cevik Onar, S., Cebi, S., Oztaysi, B., Sari, I.U., Eds.; Springer: Cham, Switzerland, Volume 504, 624–631.