Consensus Protocol Security Analysis Using an Algebraic Virtual Machine

Oleksandr Letychevskyi[a], Volodymyr Peschanenko[b], Sergiy Horbatiuk[a]

[a] *Glushkov Institute of Cybernetics of NASU, 40, prospect Glushkova, Kyiv, 03187, Ukraine*
[b] *Kherson State University, 27, Universitetska st., Kherson, 73000, Ukraine*

Abstract

The paper presents the behavior algebra formal methods for application within the consensus protocol security analysis domain in the scope of the program system AVM (Algebraic Virtual Machine). It was developed on the basis of behavior algebra and the theory of interaction of agents and environments. AVM allows the use of different methods for analysis of system properties, detection of the behavior given as algebraic patterns, and symbolic modeling of system behavior. The simplified version of Proof of Stake protocol is considered as the example of application of the algebraic approach for resolving the problem of checking resistance to attacks and reachability of undesirable properties. The study presents formalization of the protocol and its properties as behavior algebra equations. Methods of property reachability and analysis in the scope of behavior algebra such as symbolic modelling and algebraic matching are considered. The works are in progress and the study demonstrates the first applications of the approach.

Keywords
Keywords
Algebraic modeling, consensus protocol, formal methods, cyberattack, symbolic modeling, blockchain

1. Introduction and Problem Definition

Each consensus protocol provides the ability to counter malicious acts under certain conditions. For example, an attack such as double spending in the Proof of Stake protocol cannot succeed if the attackers do not own 51% of the nodes. With the emergence of new consensus protocols and their forks, resistance to such attacks may change, as well as the conditions of their implementation. Also of great importance is the reachability of undesirable properties of protocols, such as centralization. Many studies have described the use of probabilistic approaches and imitation modeling for analysis of such problems.

Our approach involves the use of an algebraic approach in studying the reachability of the properties and conditions of various attacks using a so-called behavioral algebra and its formal methods.

This study uses algebraic specifications and formal methods based on algebraic modeling and automatic proof of statements. The first experiments with consensus protocols, such as Proof of Stake and Proof of Delegated Stake, have yielded some results in assessing centralization and studying the conditions of some attacks. Therefore, further research envisages progress towards the formalization and research of other protocols, including those that are emerging in modern blockchain-based technology.

To study these properties, a so-called algebraic server was used, which contains a fairly large set of methods based on the algebra of behavior, the theory of interaction between agents and environments,

IntelITSIS'2022: 3rd International Workshop on Intelligent Information Technologies and Systems of Information Security, March 23–25, 2022, Khmelnytskyi, Ukraine
EMAIL: oleksandr.letychevskyi@litsoft.com.ua (O. Letychevskyi); volodymyr.peschanenko@ litsoft.com.ua (V. Peschanenko); gorbatiuk_sergiy@i.ua (S. Horbatiuk)
ORCID: 0000-0003-0856-9771 (O. Letychevskyi); 0000-0003-1013-9877 (V. Peschanenko); 0000-0001-6834-4211 (S. Horbatiuk)

and the theory of predicate transformers. In this algebraic server, which we call an algebraic virtual machine (AVM), the appropriate method is selected depending on the task class.

Our approach involves two components of research:

- Formalization of knowledge, namely the presentation of a consensus protocol in the form of specifications of algebra behavior. This includes the formalization of the properties to be verified, as well as the formalization of the attacker's behavior.

- The use of formal algebraic server methods to solve problems of reachability of properties and search for vulnerabilities to attack behavior.

By using this approach, we are able to identify the behavior of the system leading to the desired property, or an example of a scenario leading to the state when the attack took place.

2. Related Works

The emergence of a significant number of consensus protocols has led to a large number of papers dedicated to security analysis and countering attacks by intruders in the blockchain network. In addition, the study of such properties as the protocol's tendency towards centralization is of considerable interest to DLT-based platform developers.

Despite the innovativeness, probability approach and non-determinism of modern proof-based consensus protocols, they are a source of vulnerability to intruders. Attackers use forking in blockchain and the fact that they interact mostly through available peer-to-peer networks. Each protocol has a certain resistance to attacks when the attacker does not act according to the rules of the protocol. The main types of attacks in the protocols analyzed in the literature are double spending, consensus attack, network attack.

The main method of analyzing the security of consensus protocols is the use of probability models. Bitcoin offers a blockchain structure that publicly stores the entire transaction history as a distributed and ordered to prevent double spending [1].

In [2] it was shown that a double-spending attack is possible under certain conditions in the bitcoin network, in particular in the presence of 51% of attackers in the network. A more meaningful study of the semantics of double spending attack is considered in [3],[4].

In general the 51% attack is a problem for other consensus protocols. Although the double-spending attacks were never carried out on the Bitcoin network, the mining pool is already divided among a small number of stakeholders. In this case, there is a problem of centralization, which can completely destroy the consensus mechanism by the main owners of the mining pool.

Selfish mining attack is considered in [5] in the context of Proof-of-Work. An attacker with a mining power of less than 51% of the network can create new blocks without using them according to the rules of consensus. He sends them later, creating forks that are longer than honest miners chain. In this way it is possible to provide double-spending attack if you own 25% of the network.

A type of selfish mining attack is a “block discarding” attack [6]. In this case, the nodes of the network attacker give up blocks that come from honest members of the network and publish only the blocks that have passed from controlled nodes. The delay of the spread of honest blocks makes possible the conducting of double-spending attacks.

The eclipse attack [7] is another way to control information and influence honest users. The topology of the blockchain network may consist of regions that have a certain number of nodes. When attacking eclipse, the attacker builds a fictitious network around an honest node and controls all the information received from him by controlling the creation of the blockchain. Quite a detailed safety analysis was made in [8].

Another Proof-of-Stake protocol has some advantages over Proof-of-Work, especially saving energy costs for mining, but it introduces new vulnerabilities that increase the centralization issue. For example, in the first versions of Proof-of-Take, there was a “nothing at stake” attack. Such an attack is analyzed in [9].

In [10] the exploitation of another vulnerability Proof-of-Stake is considered - "long-range attack". This attack uses the ability to overwrite old network blocks that have already been accepted by participants.

Another Proof-of-Elapsed Time protocol is based on the probable time allocation for activating validators and creating blocks. In [11] the vulnerabilities of the applied systems that control a random time algorithm are analyzed, and in [12] it is shown that an attacker who controls part of the nodes can also control the consensus.

In the Proof-of-Burning protocol [13] the right to create blocks and be a validator is defined of burned cryptocurrency. But here as well it is possible to exploit the creation of forks to cancel transactions with burning and restore the assets.

Vulnerabilities in the Proof-of-Authority protocol related to synchronization are discussed in [14].

Proof-of-Delegated Stake has an advantage over Proof-of-Stake and solves certain security issues. The peculiarity of the protocol is that the functions of validators are performed by a small number of elected delegates. But here as well it is possible to conduct attacks of double-spending, as this is enough for some of them to conspire. Analysis and possible solution of problems are considered in [15].

Thus, the problems of malicious behavior have not yet been fully resolved in any of the consensus protocols. The search for approaches to analysis and finding effective countermeasures continues. Although the study is dominated by a probability approach, other methods and theories can also be useful and effective. One such method is the algebraic approach proposed in the article.

3. Theoretical Background

Interaction in consensus algorithms involves the participation of many entities, which are blockchain nodes that perform their function as validators or recipients of blocks. To formalize their interaction, we used the theory of agents and environments, which was created by Letychevsky (Senior) and Hilbert in the 1990s [16].

An agent is an entity that has attributes and behaviors represented by a set of its states. The environment allows agents to interact with each other and is also an entity that has attributes and can perform certain actions. The function of the insertion of the agent in the environment, which is represented by the equations of algebra of behaviors and the set of possible actions of agents, is determined.

Behavior algebra [17] is a two-sort algebra with operations "." - prefixing, "+" - nondeterministic choice and a set of terminal constants. The main sort is a set of behaviors, and the second sort is a set of actions. The terminal constants are successful termination Δ, deadlock 0, and divergent behavior $\perp$. The approximation relation $\sqsubseteq$ is a partial order on the set of behaviors with minimal element $\perp$. Operations are defined over a set of actions and behaviors.

The prefix $a.B$ means that action a precedes behavior B, and nondeterministic choice determines the branching of behavior. Behaviors are represented by equations of behavior that include the name of the behavior on the left and a behavioral expression on the right that is created from actions and other behaviors with corresponding operations on them.

In addition, algebra is extended by operations of sequential (“;”) and parallel (“||”) compositions, which, in turn, are expressed through prefixing and nondeterministic selection.

Examples of behavior expressions include the following:

$B0 = a1.a2.B1 + a3.B2,$

$B1 = a4.\ \Delta,$

$B2 = \ldots$

These imply that behavior *B0* can be interpreted as a sequence of actions *a1* and *a2*, followed by behavior *B1*, or as action *a3*, followed by behavior *B2*. Behavior *B1* will finish after action *a4*.

The language of action involves the use of various theories.

The agent and the environment can have typed attributes, for example, integer, real, list, symbolic, and enumerated.

The initial state of the agent system is specified in the form of a formula over the attributes of agents and the environment.

Each action can define expressions over the attributes of the environment and agents.

The action is presented in the form of a triple, which includes the precondition, postcondition, and process components of the action.

The precondition contains a predicate in first-order logic, which may include quantifiers.

The predicate contains expressions with predicate symbols of the corresponding theories and symbols of relations between attributes.

Postcondition contains the changing of the state of the system after the action, which is also given by formulas or operations in the relevant theories.

The process component is intended to illustrate the execution of the action and may contain any symbolic information.

For example, the expression

(a.x >= 1) ∨ ~((Head(a.list) != 0) ∧ (a.arrow == UP))

combines the theory of integers, lists, and enumeration types in the expression of first-order logic over the corresponding attributes of agent a.

Formal methods involve transforming and finding solutions to behavioral equations. One of the transformations is unfolding the behavioral expression as well as bringing it to the canonical form. Equations can have unknown behaviors.

The search for an unknown behavior that matches a pattern is done by unfolding methods with different strategies.

When unfolding behavioral equations, scenarios emerge in the form of a sequence of actions that lead to the solution of an equation or state in which the desired property is reachable.

We define the *symbolic modeling* of such a scenario in terms of the algebra of behaviors as follows:

1. Checking the satisfiability of the formula of the precondition of action and the current environment; and

2. Transitioning to another state according to the postcondition using the methods of the theory of predicate transformers [18].

4. Formalization of Consensus Protocol

Let us consider the formalization of consensus protocol as the behavior algebra application. The environment for blockchain nodes to interact is an entity that has attributes that are known to all agents, such as the number of time slots (TIME_SLOTES) and epochs (EPOCHES).

Agents of type NODE are nodes of a blockchain network and have attributes that fix the order of blocks and references to other blocks, the number of chains, chain lengths, and other values that are fixed in the description of the agent environment:

```
agent_types : obj (
          NODE : obj(
                fraud: int,
                ref : (int) -> int,
                block : (int, int) -> int,
                Validator : int,
                blockCreated : int,
                refCreated : int,
                blockReceived : (int) -> int,
                refReceived : (int) -> int,
                CHAINS : int,
                chainLength : (int)->int,
                maxLength : int,
                INIT : int,
                received :int,
                printed:(int)->bool,
                fraudBlock: (int)->int,
                Length : (int)->int
          )
    );
```

The environment description contains the description of agent types, names of interacting agents and the initial state of environment where attributes can be presented by concrete values or formulas.

In our approach we consider a simplified consensus algorithm.

The entire time of operation of the protocol under study is divided into time slots, which, accordingly, are divided into epochs. In each time slot, the selected agent creates a block and a link to the previous block and sends it to all other agents during the time slot.

Other agents obtain blocks and form similar blockchains in their environment. There may be time delays in the process of sending blocks, which generates forks.

In this case, the algorithm determines that the block creator must continue the longest chain and, in the case of equal lengths, the one that was created earlier in time.

At a certain chain length, finalization occurs and the blocks remaining in the non-continuing chain must be removed, which means that transactions are returned to the appropriate pool from which the new blocks are created.

Below is a system of behavioral equations that conform to the consensus protocol:

```
B0 = (BLOCK_CREATION; BLOCK_WIDESPREAD; NEXT_SLOT; B0),

BLOCK_CREATION = selectValidator. createBlock. (
     (fraudFeature.VALIDATOR_FRAUD) +
     (not_fraudFeature. VALIDATOR_NOT_FRAUD); sendBlock),

VALIDATOR_FRAUD = (createRefFraud +
      (createRefNormalFraud; fraudEpoch1 + fraudEpochOther)),

VALIDATOR_NOT_FRAUD = createRefNormal,

BLOCK_WIDESPREAD =
((receiveBlocks.(insertBlocks + not_insertBlocks);
         insertForks + not_insertForks;
         recalcMaxLength + !recalcMaxLength)    +
         not_receiveBlocks;
         signalDelay + not_signalDelay),

NEXT_SLOT = nextSlot + nextEpoch + lastSlot.Delta
```

Equations contain behaviors that are expressed through actions and other behaviors.

The behavior B0 consists of three sequential behaviors `BLOCK_CREATION`, `BLOCK_WIDESPREAD`, and `NEXT_SLOT`. `BLOCK_CREATION` selects the validator which creates the block and sends it to all other agents. `BLOCK_WIDESPREAD` is the behavior where all other agents receive the blocks and insert it in their blockchain correspondingly to the reference. `NEXT_SLOT` checks the end of the epoch and defines the next slot. Afterward these behaviors continue in the cycle.

Next, we define the actions that appear in these behaviors. To illustrate the method, the following most important actions in the consensus algorithm are provided, while other technical and support actions are omitted.

```
  selectValidator = Exist(i : NODE)(agentNODEID(i) == timeSlot) ->
  (i.Validator = 1),
  createBlock = Exist (i:NODE) (i.Validator == 1) ->
  (BlockNumber = BlockNumber + 1; i.blockCreated = BlockNumber),
  createRefNormal = Exist (i:NODE, j : int) (i.Validator == 1 &&
   i.maxLength == i.chainLength(j) && 1 <= j <= 20 && j<= i.CHAINS)
->
  (
     i.refCreated = i.block(j,i.chainLength(j));
```

```
      i.block(j,i.chainLength(j) + 1) = i.blockCreated;
      i.ref(i.blockCreated) =  i.block(j,i.chainLength(j));
      i.chainLength(j) = i.chainLength(j) + 1;
      i.Length(i.blockCreated) = i.chainLength(j) + 1;
      delay(i.blockCreated) = -1
  ),
  sendBlock = Exist (k:NODE) Forall(i : NODE) (k.Validator == 1 &&
i.Validator != 1) -> 1,
  receiveBlocks = Exist (x: int, y: int, z: int) Forall (i: NODE, k:
NODE) (delay(x) <= 0) ->
   (i.refReceived(i.received + 1) = y; i.blockReceived(i.received + 1)
= x;
    i.received = i.received + 1; i.fraudBlock(x) = z),
  insertBlocks = Forall (i : NODE, j : int, k : int)
   (1 <= j <= i.CHAINS && 1 <= k <= i.received &&
    i.refReceived(k)      ==      i.block(j,i.chainLength(j))      &&
delay(i.blockReceived(k)) <= 0  &&
  i.refReceived(k) >= 0) ->
   (i.block(j,i.chainLength(j) + 1) = i.blockReceived(k);
  i.chainLength(j) = i.chainLength(j) + 1;
  i.ref(i.blockReceived(k)) = i.refReceived(k)),
  fraudFeature = Exist (i:NODE) (i.Validator == 1 && i.fraud == 1) -
> 1,
  not_fraudFeature = Exist (i:NODE) (i.Validator == 1 && i.fraud ==
0) ->1,
  fraudFeature = Exist (i:NODE) (i.Validator == 1 && i.fraud == 1) -
> 1,
  not_fraudFeature = Exist (i:NODE) (i.Validator == 1 && i.fraud ==
0) ->1,
  insertForks = Forall (i : NODE, j : int, k :int)(
      1 <= j <= i.CHAINS && 1 <= k <= i.received &&
      i.refReceived(k) != i.block(j,i.chainLength(j)) &&
      delay(i.blockReceived(k)) <= 0 && i.refReceived(k) >= 0) ->
  (i.CHAINS   =   i.CHAINS   +   1;   i.block(i.CHAINS   +   1,1)   =
i.blockReceived(k);
      i.ref(i.blockReceived(k))          =          i.refReceived(k);
i.chainLength(i.CHAINS)      =      i.Length(i.refReceived(k))      +
1;i.Length(i.blockReceived(k))  =  i.Length(i.refReceived(k))  +  1;
i.refReceived(k) = -1)
```

The actions contain universal and existence quantifiers in precondition. Every action changes the environment of blockchain if the precondition is satisfiable.

This algorithm is a simplified version of existing Proof of Stake consensus algorithms. In it, however, some parts of the functionalities remained out of scope.

The purpose of this presentation is to show a method that can be extended later, considering all the other features of the algorithms.

5. Formalization of Properties

Using formal methods within the algebra of behaviors it is possible to solve the problem of the reachability of a property. In this case it is necessary to make a formula that expresses this property. One such formula can be a condition of operation of a double-spending attack.

We have simplified this condition somewhat, given that the main purpose is to demonstrate the method.

Let's define a part of the agents creating blocks as malicious.

Such agents will create blocks and links to previous blocks to further support not the longest chain, but smaller ones. In addition, to create alternative circuits, the attacker simulates a time delay by sending the block in subsequent time slots.

In this case, the finalization will be delayed, and a double-spending attack is possible, because the finalization is triggered at a certain length of the longest chain. This length determines the number of blocks to be finalized (*FinalizationValue*).

The attacker's action to create a link to the previous block will be as follows:

```
createRefFraud = Exist (i:NODE, j : int) (
 i.Validator == 1 &&
 i.maxLength > i.chainLength(j) &&
 1 <= j <= 20 && j<= i.CHAINS) ->
(
    i.refCreated = i.block(j,i.chainLength(j));
    i.block(j,i.chainLength(j) + 1) = i.blockCreated;
    i.ref(i.blockCreated) =  i.block(j,i.chainLength(j));
    i.chainLength(j) = i.chainLength(j) + 1;
    i.Length(i.blockCreated) = i.chainLength(j) + 1;
    delay(i.blockCreated) = -1
    )
```

The double-spending formula then determines the sum of the lengths of all chains formed after the last finalization.

$$\forall (i, 1<=i<=CHAINS) \qquad \Sigma\, a.chainLength(i) > FinalizationValue$$

6. Search for Reachability by Symbolic Modeling

If we know the relationship between the number of malicious and honest agents, then we can perform symbolic modeling based on this relationship. The first method is forward symbolic modelling of the protocol in order to reach this formula.

This method can be used when the property is reachable, for example, with most attackers. However, if the property is unreachable, then due to the exponential explosion of system behavior scenarios, we may not reach the end, even using the properties of the equivalent states and symmetries in the search algorithm.

Various targeted search methods, including heuristics, can help but do not guarantee an exhausted search of all scenarios.

Another method used to find the reachability of a property is backward symbolic modeling, where the search is conducted from the state represented by the operation of the successful attack formula to the initial state.

In this case, it is possible to bypass all reachable states, which may be much smaller than the total number of states. A complete bypass, leading from a state of successful attack, can end in a subset of behavioral equations.

Scenarios that occur during modelling are symbolic, i.e., with arbitrary attribute values.

A concretization program that is a part of an algebraic virtual machine creates a scenario or a trace with specific attributes that match the symbolic trace or the sequence of states of an agent's environment.

7. Search for Behavior by the Methods of Algebraic Matching

Within the algebra of behaviors there are other methods that can solve the problem of reachability of a property, such as the method of algebraic matching.

In addition to defining a property, we can specify known facts in behavior that can lead to that property.

In this case, we can determine the sequence of actions during the period starting from the event of finalization.

Known actions of malicious agents could be, for example, the continuation of a chain that is not the longest.

Therefore, we can define such behavior, and the limitations for it, as follows:

$X = finalization.Y,$

$Y = createRefFraud.Y$

$\forall (ai, Y = a1,a2,\ldots)\ !(ai = finalization)$

where *createRefFraud* is the action of malicious agents that violate the consensus rules and do not continue the longest chain.

This behavior is a cycle that begins with the finalization event and contains a sequence of malicious actions. Reducing the search eliminates the need for full unfolding.

By having such a description of behavior, we can solve the behavioral equation using algebraic matching. This method consists of two stages.

The first is unfolding the consensus algorithm according to a sequence of actions. If such an unfolding exists, we can perform symbolic modeling according to the scenario of this unfolding.

We can then perform an algebraic matching of the properties that appear in the preconditions of the actions of unknown behavior with the states of the system in the modeling.

Algebraic matching is the determination of the satisfiability of the conjunction of agent states and the preconditions of the actions of unknown behavior.

When finding a trace with symbolic attributes, it is possible to build, as a counterexample, a scenario of behavior with concrete attributes.

8. Experiment Results and Further Work

Experiments were performed for a simplified consensus algorithm corresponding to Proof of Stake. Formulas were determined to represent a successful double-spending and Sybil attack.

The proof was performed using the method of backward symbolic modeling and algebraic matching for large number of agents.

The work is currently underway to prove these properties for an arbitrary number of agents. To do this, we plan to use the properties of quantifiers in the theory of predicate transformers [16] and the method of mathematical induction created for behavioral algebra.

The analysis and proof of properties is being performed on the algebraic server that is a part of the algebraic virtual machine (AVM). It is a platform for the application of algebraic methods in analyzing formal specifications.

There are a number of methods that have been developed during the last two decades within the scope of behavior algebra, including the property reachability definition methods.

These are based on a number of approaches, including symbolic modeling (forward and backward), slicing dependency analysis, invariant generation and other relevant methods. AVM includes methods of algebraic matching of behavior.

This involves matching a system's behavior with patterns of behavior. AVM also has scenario generation subsystem.

This method includes the generation of scenarios with required coverage of the model for further black-box testing that is implemented by symbolic modeling methods.

The algebraic server allows for conducting static proving of statements like the safety properties, which are stored in the database or user-defined properties; classical properties like inconsistency, incompleteness; annotation statements; time properties; and other domain-specific properties.

Analysis of properties of consensus algorithm was successfully implemented on the AVM for large number of nodes.

The experiment was launched for different number of fraud nodes. Assuming 50% of frauds the backward modeling did not reach the initial state and finished the exhausted search within the restricted area of search space.

When we defined 51% of frauds, we reach the double-spending attack property and created the scenario that leads to this state.

In addition, it is possible to graphically visualize the state in which the successful attack took place, in the case where most agents are attackers. Figure 1. shows the state of a successful double-spending attack.

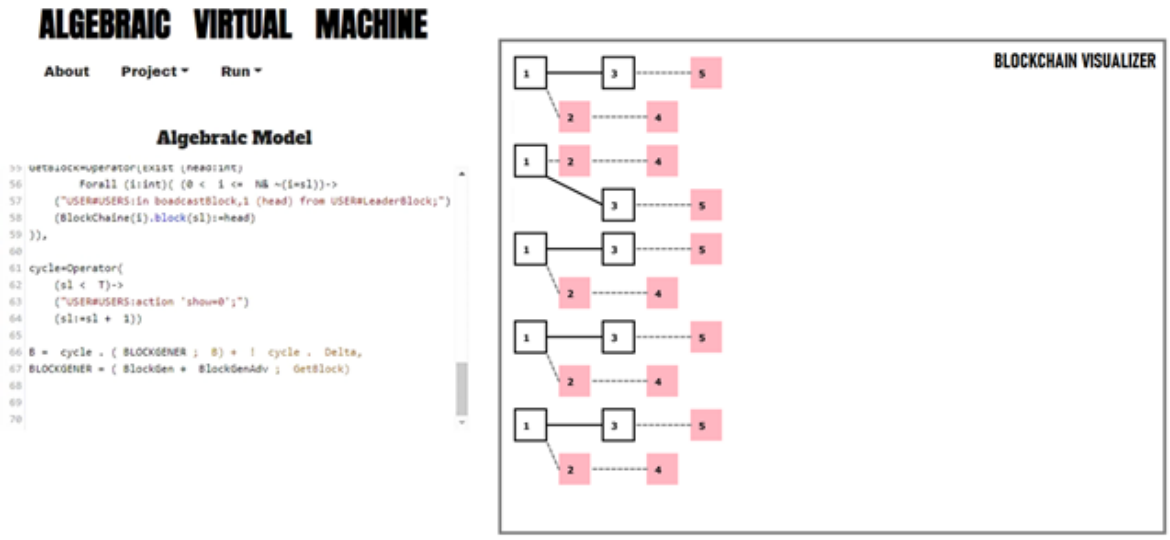

Figure 1: Algebraic Virtual Machine: Double-spending attack

We are planning further research with more detailed consensus protocols.

In particular, our research continues on the Proof of Delegated Stake consensus protocol in solving the problem of the reachability of the centralization property, i.e. the state of the system when most resources are captured by a small number of agents.

We will also expand the set of attacks to be formalized in terms of the algebra of behaviors for various other consensus algorithms.

9. Conclusion

This paper considers an algebraic approach to the analysis of the properties of the consensus protocol. The difference of this approach from the existing methods of probabilistic and imitation modeling is that in the algorithms of verification of reachability there is no convergence, which can lead to inaccurate results.

On the other hand, in the algebraic approach there is a difficulty in solving the problem of satisfiability of formulas.

Although modern solvers are quite powerful, the emergence of nonlinear arithmetic formulas adds complexity and the problem may not be solved within this formalization. One way out is to approximate formulas and use known numerical methods.

The algebraic server we use contains a library of formal methods implemented within the framework of behavioral algebra. Adding heuristics and allocating model classes will give a very powerful gain in the analysis of algorithms and protocols.

10.References

[1] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system (2008). URL https://bitcoin.org/bitcoin.pdf. Last access: 15 March 2021.

[2] G.O.Karame, E. Androulaki, S. Capkun. Double-spending fast payments in bitcoin. In: ACM CCS 2012, pp. 906–917 (2012).

[3] M. Karpinski, L. Kovalchuk, R. Kochan, R. Oliynykov, M. Rodinko, L. Wieclaw. Blockchain Technologies: Probability of Double-Spend Attack on a Proof-of-Stake Consensus. Sensors 2021, 21, 6408.
[4] C. Pinzón, C.Rocha. Double-Spend attack models with time advantange for bitcoin. Electron. Notes Theor. Comput. Sci. 2016, 329, 79–103.
[5] I. Eyal, E.G. Sirer. Majority is Not Enough: Bitcoin Mining is Vulnerable. Commun. ACM 61(7), 95–102 (2018). DOI 10.1145/3212998. URL http://doi.acm.org/10.1145/3212998
[6] L. Bahack. Theoretical Bitcoin attacks with less than half of the computational power (draft). arXiv preprint arXiv:1312.7013 (2013).
[7] E. Heilman, A. Kendler, A. Zohar, S. Goldberg. Eclipse attacks on bitcoin's peer-to-peer network. In: USENIX Security'15, pp. 129–144 (2015).
[8] Ren Ling. Analysis of Nakamoto Consensus. Cryptology ePrint Archive, Report 2019/943. (2019). https://eprint.iacr.org/2019/943.
[9] Rebello, G.A.F., Camilo, G.F., Guimarães, L.C.B., de Souza, L.A.C., Duarte, O.C.M.B.: On the security and performance of proof-based consensus protocols. In: 2020 4th Conference on Cloud and Internet of Things (CIoT), pp. 67–74 (2020). DOI 10.1109/CIoT50 422.2020.9244295.
[10] E. Deirmentzoglou, G. Papakyriakopoulos, C. Patsakis. A survey on long-range attacks for proof of stake protocols. IEEE Access 7, 28712–28725 (2019).
[11] S. van Schaik, A. Kwong, D. Genkin, Y.Yarom. SGAxe: How SGX fails in practice (2020)
[12] L. Chen, L. Xu, N. Shah, Z. Gao, Y. Lu, W. Shi. On security analysis of proof-of-elapsed-time (PoET). In: International Symposium on Stabilization, Safety, and Security, pp. 282–297. Springer (2017).
[13] Iain Stewart: Proof of Burn (2012). URL https://en.bitcoin.it/wiki/Proof_of_burn. Last access: 15 March 2021.
[14] S.D. Angelis, L. Aniello, R. Baldoni, F. Lombardi, A. Margheri, V., Sassone. PBFT vs proof-of-authority: applying the CAP theorem to permissioned blockchain. In: Italian Conference on Cyber Security (06/02/18) (2018). URL https://eprints.soton.ac.uk/415083/
[15] Y.Gilad, R. Hemo, S. Micali, G. Vlachos, N. Zeldovich,: Algorand: Scaling byzantine agreements for cryptocurrencies. In: Proceedings of the 26th Symposium on Operating Systems Principles, pp. 51–68 (2017).
[16] D. Gilbert, A. Letichevsky. A model for interaction of agents and environments, in D. Bert, and C. Choppy, eds., Recent Trends in Algebraic Development Techniques, LNCS 1827, Springer-Verlag, Cham, Switzerland, 1999.
[17] A. Letichevsky, Algebra of behavior transformations and its applications, in V. B. Kudryavtsev and I. G. Rosenberg, eds., Structural Theory of Automata, Semigroups, and Universal Algebra, NATO Science Series II. Mathematics, Physics and Chemistry - vol. 207, pp. 241-272, Springer, 2005.
[18] A. Letichevsky, O. Letychevskyi, V. Peschanenko, and T. Weigert, Insertion modeling and symbolic verification of large systems, in J. Fischer, M.Scheidgen, I. Schieferdecker, and R. Reed, eds., SDL 2015: Model-Driven Engineering for Smart Cities, pp. 3-18, Cham, Switzerland: Springer International Publishing, 2015.