Multidimensional Blockchain as Robust Distributed Ledger

Ilya Shilov, Danil Zakoldaev
ITMO University
St. Petersburg, Russia
ilia.shilov,d.zakoldaev@itmo.ru

***Abstract*—The paper observes the problem of scaling robust distributed ledgers based on blockchain technology. A solution for the problem is presented - multidimensional blockchain. A model for protocol implementing ideal multidimensional blockchain is constructed. It is proven that multidimensional blockchain and its corresponding protocol GUC-implement robust distributed ledger.**

I. Introduction

Multidimensional blockchain is a young technology that largely borrows the features of a conventional one-dimensional blockchain used to build modern distributed applications like cryptocurrencies. Multidimensional blockchain has two distinctive features: an internal data structure that allows to include a set of blockchains and search and verification protocol. Structurally, a multidimensional blockchain consists of an arbitrary number of blockchains, when each blockchain, but the first one, goes through the registration procedure in one of the existing blockchains. Further inter-system interaction is performed using search and verification protocol.

Multidimensional blockchain has been developed as a method for inter-system exchange for robust distributed ledgers based on blockchain technology. Therefore, previous analysis of multidimensional blockchain security assessed how the fact that the blockchain was placed into multidimensional blockchain and used the functionality of external transactions impacted security. A security proof has been presented that multidimensional blockchain had not broken the security of internal one-dimensional blockchains (if security constraints were followed) [1].

However, another important direction for multidimensional blockchain exists: it can be used to scale robust distributed ledgers, which appears to be among the most important problems for blockchain technologies [2]. But in order to use it for such goal it is necessary to prove that multidimensional blockchain implements robust distributed ledger itself. Previously the security of multidimensional blockchain itself as a separate system has not been considered separately. The purpose of this work is to build such a proof.

To achieve this goal, several tasks are being solved. In particular, one of the shortcomings of previously published works is eliminated – a simplified representation of a multidimensional blockchain model, which does not take into account the developed models of a robust distributed ledger, as well as the results achieved in the field of data exchange between independent stable distributed ledgers. In addition, the MBC-Protocol is formally introduced and models for the proof are built. Finally, with a simulator and the universal composition theorem it is proven that multidimensional blockchain independently implements a robust distributed ledger, i.e. can be used to securely scale robust distributed technologies.

II. Previous work review

Multidimensional blockchain has been introduced in 2020 in [3]. In this paper, the main aspects of the technology implementation have been considered and a comparison has been made with existing alternatives in terms of the availability and integrity of the information being processed. It has been shown that multidimensional blockchain could exist either in state or block model. Next the pseudo-algorithms for its functioning and inter-system exchange have been presented. The analysis has been based on the constraints and principles of Bitcoin [4].

A formal security proof for multidimensional blockchain has been proposed in [1]. It was based on various methods of the theory of probability, as well as the UC-Framework (Universal Composability Framework). This framework is used to prove the security of protocols. It implies building models of protocol implementations with interactive Turing machines (Turing machines able to communicate). It represents both target functionality (protocol) and the implementation which security is under consideration. Generalized version of the framework (GUC-Framework) differs in that it allows using common functionalities for protocols running simultaneously. The proof requires either of:

1) Showing that for any adversary attacking the target system there exists a simulator attacking the implemented system (or vice versa) such that external observer (environment) is unable to distinguish the executions. Sometimes it is necessary to show that ”bad” events happen with negligible probability. Bad events mean events which prevent the simulation.
2) Building a sequence of hybrid models – models in which some parts are sequentially replaced to create a sequence of equivalent models from the target to the implementation (or vice versa).
3) Showing that the target model and implementation are equivalent using theory of probabilities and describing

ISSN 2305-7254

every possible input and output of these models (i.e. the framework is used only for formalization).

Using universal composability theorem which allows to replace parts of models with equivalent ideal functionalities (e.g. digital signature or network) simplifies both proof constructions.

At the same time, in [1], only a heuristic proof of multidimensional blockchain security has been proposed. It has been shown that multidimensional blockchain GUC-implements a robust distributed ledger (by GUC-implementation we mean the equivalence of the model execution in case of proposed technology and corresponding ideal functionality). Moreover, to prove one of the statements, the universal composition theorem has been used, which is not entirely correct given the absence of a formal GUC security proof for some of the functionalities among the replaced objects in hybrid proof. The paper mentions multidimensional blockchain protocol (MBC-Protocol), but does not properly represent it.

Also, the security proof [1] does not take into account an analysis of inter-system exchange security. It is based on various previously published works like [5]. Some analysis of inter-system exchange had been created before, for instance, in [6]. But for multidimensional blockchain a specially created search and verification protocol for blocks and transactions has been created [7]. A thorough analysis of various possible construction for it have also been presented in that paper.

Next on the basis of robust distributed ledger models – Ouroboros [8] and Bitcoin [9] – an adapted model of robust distributed ledger has been built in [10]. This model supports the functionality of a multidimensional blockchain, i.e. external transactions.

III. MBC-Protocol

To build a complete proof of security for multidimensional blockchain, it is required to formalize the MBC-Protocol model. Some aspects of its construction were previously presented in [1]. However, the protocol itself was not given. This protocol is understood as a program executed by Interactive Turing Machines (ITM) when simulating a system of devices that jointly support a robust distributed ledger based on a multidimensional blockchain. In addition, the GUC-model of the ideal functionality that implements the multidimensional blockchain has not been presented before.

In general, the system is a function of transition between states [11]:

$$\sigma_{t+1} \equiv \Pi(\sigma_t, B) \mid B \equiv (D, (T_i)_{i=1,...N}), \tag{1}$$

where σ_t is the t-th state of the system, Π is a state transition function, B is a block that contains transactions and additional system information. The block itself consists of auxiliary information D and a set of transactions T_i. The state structure is defined by an application and an available set of transactions. The transition between states is performed using function Π:

$$\Pi(\sigma, B) \equiv \Omega(\Upsilon(\Upsilon(...(\Upsilon(\sigma, T_0), ...), T_n))), \tag{2}$$

where Υ is the function of transition between system states and Ω is finalizing function. In other words, block-level state transition is performed with sequential applying of transactions with state-level functions and construction of block is finished with finalizing function (represents consensus mechanism).

This model satisfies most of the currently used blockchains, although they differ in the structure of the B block, the finalizing function, the set of transactions that cause the system to transition from state to state, and the state transition function.

Let's move on to considering a mathematical model of a multidimensional blockchain. Since blockchains create new states at different rates, the following relationships assume that transactions were created over a fixed length of time - a slot. For the most correct formulation of the mathematical model, the following ratios can be adopted:

$$sl \equiv GCD(Time(\sigma_t^{(k)} \rightarrow \sigma_{t+1}^{(k)})), \tag{3}$$

where sl is slot (a period of time), GCD is greatest common divider function, $Time$ is a function which returns the duration of state transition. As multidimensional blockchain contains multiple conventional blockchains each slot is associated with a large set of transactions. In one-dimensional blockchain model there is only one blockchain. The connection between these models is following:

$$T^k = (T^{(k,1)}, ..., T^{(k,j)}) \mid j = [\frac{Time(\sigma_t^{(k)} \rightarrow \sigma_{t+1}^{(k)})}{sl}], \tag{4}$$

where T^k is a set of transactions in one round of separate conventional blockchain, $T^(k, i)$ is a set of transactions in conventional blockchain inside multidimensional blockchain, j is the number of slot. For simplicity the set of transactions in round j for conventional blockchain k in multidimensional blockchain can be shown as:

$$T^{(k,j)} = (T_0^{(k,j)}, ..., T_n^{(k,j)}), \tag{5}$$

where n is the number of transactions.

In other words, a slot is the largest period of time by which the time intervals required for the transition between states in all blockchains are divided. In practice it can be taken as the least discrete period of time. As a result, each transition between states in each blockchain is carried out once in a fixed (integer) number of slots. That is:

$$\Pi'(\sigma^k, T^{(k,j)}) = \Omega(\Upsilon(...\Upsilon(\sigma^{(k)}, T_0^{(k,j)}), ..., T_n^{(k,j)})) \tag{6}$$

Or in case the set of transactions is empty:

$$\Pi'(\sigma^k, \varnothing) = \sigma^{(k)} \tag{7}$$

A similar model is used to prove the correctness of proof of stake protocols [8]. In the given model, it is believed that in some blockchains, blocks are not created in every time slot. However, it is important that the concept of a slot used in this paper may differ from the concept of a slot used in other papers on security of consensus mechanisms.

In general, multidimensional blockchain is a transition function over compound system state:

$$\Sigma_{i+1} \equiv \Phi(\Sigma_i, T) \mid \Sigma_i \equiv \{\sigma^{(1)}, ..., \sigma^{(N)}\}, \quad (8)$$

where Σ_i is a compound state (state of multidimensional blockchain) and Φ is a transition function, which can be further presented as:

$$\Phi(\Sigma_i, T) \equiv \Psi(P(\Sigma_i, T), T), \quad (9)$$

where Ψ - creates new blockchains, P is the state transition function.

The state transition is a sequential state transition for each conventional blockchain inside the multidimensional one:

$$P(\Sigma_{i+1}, T) \equiv E(E(...E(\Sigma_i, T, 1), ...)T, K), \quad (10)$$

where E is the state transition function for the k-th blockchain in its composition, K is the number of blockchains.

To connect the novel model with conventional one [11] it is necessary to have a supplementary function:

$$E(\Sigma_i, T, k) = E'(\Pi'(\sigma^{(k)}, T)), \quad (11)$$

where E' is an auxiliary function that returns a multidimensional blockchain for a one-dimensional blockchain and is used to avoid the use of the universality quantifier in mathematical notation.

The state of a multidimensional blockchain at any given time is a set of states of all individual blockchains. In this case, state transition function Φ solves two problems: it applies transactions to individual states of blockchains and creates new blockchains, that is, it initializes their first state (genesis block). From a formal point of view, it would be correct to use addresses instead of blockchain numbers, but for simplicity, integer numbering is used.

Finally, it is required to define the relationship between the ledger state transition functions:

$$\Pi(\sigma^{(k)}, T^{(k)}) \equiv \Pi'(\Pi'(...\Pi'(\sigma^{(k)}, T^{(k,1)}), ...), T^{(k,j)}) \quad (12)$$

Therefore, multidimensional blockchain is an analogue of a conventional one-dimensional blockchain, with the difference that it is comprised of a set of ordinary robust distributed ledgers, for which the multidimensional blockchain provides an addressing abstraction.

The model of a multidimensional blockchain consists of several ITM: A – adversary, Z – environment, P_i – a node running the protocol, G_{MBC} – multidimensional blockchain ideal functionality operating according to the mathematical model given above, G_{Clock} – clock functionality. The structure of a node implementing multidimensional blockchain has been previously presented in [1], and its simplified representation is shown at Fig. 1.

Core principles of this model functioning:

1) Each node and account is characterized by a hierarchical address that determines their location in the multidimensional blockchain [3].

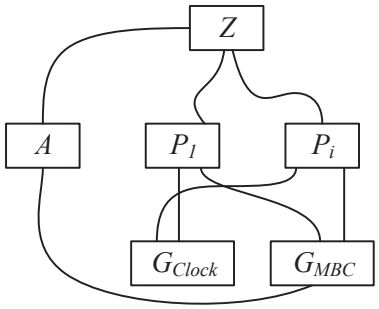

Fig. 1. GUC-model of multidimensional blockchain ideal functionality

2) Many robust distributed ledgers (G_{Ledger}) operate within the multidimensional blockchain - their code is implemented in the form of functions corresponding to [10]. A significant part is borrowed from [9] with an addition of functions and variables necessary to perform inter-system communication.
3) When an internal (ordinary) transaction is received (in a *Next-Block* message), it is applied to the ledger in a usual way, like in any conventional blockchain.
4) When an external transaction is received, it is split into two parts, each of which is added to the corresponding *Next-Block* for each participating ledger. In this case, the addition of the second part (incoming transaction) is performed with a delay Δ.

The protocol implementing this ideal functionality ($MBC-Protocol$) is presented at Fig. 2. The ITM are similar with the exception that multidimensional blockchain ideal functionality is replaced with a set of one-dimensional ideal distributed ledger functionalities (G_{Ledger}) and additional ideal functionality for search and verification is used (G_{Verify}).

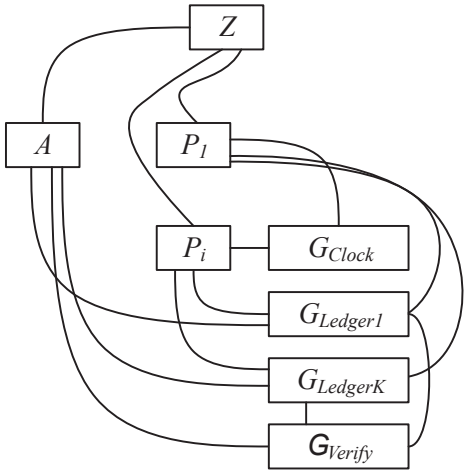

Fig. 2. GUC-model of multidimensional blockchain protocol

Proposition 1. MBC-Protocol GUC-Implements Multidimensional Blockchain.

To prove this, it is enough to show that the execution of this protocol is equivalent to the execution of the multidimen-

sional blockchain GUC-model, since in this case the universal composition theorem will be applicable. The proof is based on hybrid models, when each next model differs from the previous one, but remains equivalent to it:

- HYB0 is a multidimensional blockchain model as presented on Fig. 1. All nodes use queries to the multidimensional blockchain to work. In fact they act like "dummy" parties which only pass queries in appropriate format to an ideal functionality.
- HYB1 is a model in which nodes independently handle addressing actions, i.e. determine source and destination ledgers for each external transaction. Instead of one external transaction they redirect two internal transactions (outgoing and incoming) to the multidimensional blockchain. HYB1 is equivalent to HYB0, since the way the model uses multidimensional blockchain does not change: external transactions are simply divided in advance.
- HYB2 is a model in which nodes perform notifications on all external transactions: when a new transaction is created, a notification is sent to the nodes that maintain the target ledger. Then they independently send a request to the multidimensional blockchain. The difference from HYB1 is only in the origin of the second (incoming) transaction, since the transmission of the notification takes negligible time in the scale of the slot time.
- HYB3 is a model in which nodes independently verify an external transaction and send a request to add an incoming transaction only if it is correct. For this, the ideal search and verification functionality is used, which is guaranteed to carry out the verification correctly. The same functionality is used inside multidimensional blockchain ideal functionality. Since no changes have been made to the functionality of the multidimensional blockchain, this model is equivalent to HYB2.
- HYB4 is a model in which nodes independently carry out verification of incoming external transactions using a search and verification protocol, which must GUC-implement an ideal search and verification protocol like those presented in [7]. According to the universal composition theorem, this model is equivalent to HYB3 with a probability determined by the probability of successful verification.
- HYB5 is a model in which the multidimensional blockchain is replaced by many one-dimensional blockchains. Since verification is guaranteed (subject to the constraints of the GUC-implementation), this model is equivalent to HYB4.

It can be seen that HYB5 is the same model as given at Fig. 2. In other words, this model is actually a simulated protocol that implements a multidimensional blockchain (MBC-Protocol). Thus MBC-Protocol GUC-implements ideal multidimensional blockchain functionality and can be used in models instead of this functionality and vice versa.

IV. Proof of secure scaling robust distributed ledger with multidimensional blockchain

The MBC-Protocol, which was first mentioned in [1] but not presented, is actually defined in the previous section. A novel robust distributed ledger model with support for external transactions was presented in [10]. Any execution model in terms of a universal composition framework implies the presence of an adversary (A), environment (Z), a set of nodes (P_i) executing the target protocol, and a set of global (G) and local (F) functionalities used by them. Each of the listed entities represents an instance (ITI) of an interactive Turing machine (ITM) [12].

To prove that multidimensional blockchain GUC-implements a robust distributed ledger, it is required to show that for any attacker A in the model with multidimensional blockchain protocol there is such a simulator S for the model with robust distributed ledger (ideal functionality) that (from the view of the environment Z) the executions are statistically indistinguishable:

$$EXEC_{REAL,A,Z} \approx EXEC_{IDEAL,S,Z} \tag{13}$$

Execution means sequential activation of interactive Turing machines, starting with A, provided that a sequence of bits (input data) is transmitted to the input of each interactive machine. Then all further activations are performed either by the environment Z or by previously executed ITI via message call mechanism.

It is important to note that in current proof a dummy adversary A is used: it just transmits request from the environment to all other nodes in the execution model. The correctness of such approach was shown in [13] with a theorem which claimed that any protocol GUC-implements any other protocol if and only if it GUC-emulates it with respect to the dummy adversary.

It is also necessary to mention that the model of robust distributed ledger used here is extended with an action which was not previously taken into account in the proposed model, but necessary to build a proof. When sending a response about the result of transaction verification, the ideal search and verification functionality sends a notification to the attacker – $(VERIFY, sid, ledgerID, tx)$. This action is correct, since a node attacking the system can ask the ideal functional to verify the same transaction as the node executing the protocol.

Consider a simulator S for a node that implements a robust distributed ledger. The structure of the computational model in this case is shown in Fig. 3. The simulator internally executes a copy of a multidimensional blockchain protocol. All external messages for ITI are handled and delivered only if necessary and after processing. Actually the simulator also has to run nodes which execute multidimensional blockchain protocol, but these are not given on the schema as they are primitive nodes which only deliver messages from the environment to the ideal robust distributed ledger functionalities. In other words the simulator runs a complete copy of the emulated model.

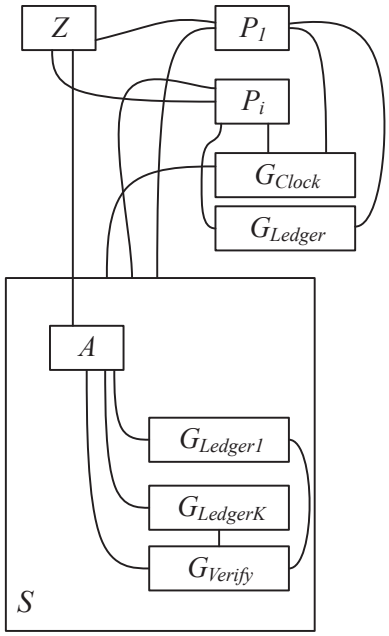

Fig. 3. GUC-model for simulator used to prove that multidimensional blockchain protocol GUC-implements robust distributed ledger

The simulator internally uses following variables:

- *ADDRESS* = {*address: ledgerID*} - a dictionary used to translate addresses of G_{Ledger} to addresses of internally executed ledgers. The translation is performed both for transactions and the parties in transactions.
- *EXTERNAL_TX* = {*ledgerID: [TX]*} - a set of external transactions delayed until their verification is ended. This is done to ensure that only external transactions with both incoming and outgoing parts applied to the multidimensional blockchain are applied to G_{Ledger}.
- *SLACK_POINTERS* = {*ledgerID: pointers*} - a structure which contains slackness pointers for all ledgers. It is necessary to store all these pointers for internal ledgers and translate them to a form appropriate for G_{Ledger}.
- *SLACK_POINTERS_ROUND* = {*ledgerID: pointers*} - a supplementary data structure to store *SET-SLACK* queries of an adversary A during round to use it as a source for the simulator's *SET-SLACK* request at the end of the round.
- *NEXT_BLOCK_TX* = {*ledgerID: [tx]*} - a data structure used to store *NEXT-BLOCK* data sent from A to simulated ledgers (which are parts of multidimensional blockchain protocol). Then these transactions are used to create the *NEXT-BLOCK* data structure for G_{Ledger}.

Also in the description of the simulator a set of procedures is used for simplification:

- **SEARCH**(*address*) $\rightarrow$ *ledgerID* - given the address of block, transaction or transaction party the function returns corresponding ledger.
- **IN**(*address*, *ledgerID*) $\rightarrow$ *boolean* - given address and ledger identifier, the function allows to check whether the node is maintaining the ledger.
- **ADDRESS**(p) - given the node, the function returns its address.

The logic for this simulator is given below. Like in [9] the simulator is described by its reactions on incoming messages and by the messages which originate from it.

On receiving messages from G_{Clock} the simulator S performs following actions:

- *(CLOCK-UPDATE, sid, p)* - a message sent by clock as notification when any node finishes its execution within round.
  1) Search: *p'* = **SEARCH**(**ADDRESS**(p)).
  2) Send a message *(CLOCK-UPDATE, sid, p')* to simulated adversary A.
  3) Remember the node from which activation was received.
  4) If this is the end of the round, then execute the **RoundEnd** procedure.

On receiving messages from G_{Ledger} the simulator S performs following actions:

- *(SUBMIT, BTX)* - a message sent by ledger after the transaction is included in the buffer of transactions available for inclusion:
  1) Find out the ledger address of the transaction origin.
  2) Send *(SUBMIT, BTX)* to corresponding ledger.
  3) Send *(SUBMIT, BTX)* to a simulated adversary A.
- *(MAINTAIN-LEDGER, sid, minerID)* - a message sent as a notification about the execution of mining operations by a node.
  1) Perform search of the miner's ledger: *p'* = **SEARCH**(**ADDRESS**(*minerID*)).
  2) Send a message *(MAINTAIN-LEDGER, sid, p')* to node A. This message is used to notify the adversary that a corrupted party has to perform consensus calculations. In fact for ideal functionalities this message has little sense but must be handled to keep the proof correctness.

On receiving messages from G_{Verify} the simulator S performs following actions:

- *(VERIFY, sid, ledgerID, tx)* - after receiving the verification result, create a *BTX* transaction and deliver it to the accepting ledger - *(SUBMIT, BTX)*.

Finally, on receiving messages internally from A the simulator performs following actions:

- *(SET-SLACK, pointers)* - the attacker configures pointers for the target ledger. Simulator has to store pointers for nodes in the *SLACK_POINTERS_ROUND* variable.
- *(NEXT-BLOCK, hFlag, (txid1, ..., txidl)* - the attacker offers the next block to create. The simulator must transfer this structure to the corresponding ledger and store the transactions in *NEXT_BLOCK_TX*.

Previously defined actions used a couple of additional procedures:

- **ReadLedger** - read the ledger: *(READ, sid, state)* $\rightarrow$ *(state, buffer, I)* - read state from the ledger.
- **RoundEnd** - end of round, the simulator sets up State-Slackness for the external G_{Ledger}.

At the end of the round, the simulator must ensure that the changes that have occurred within the simulated multidimensional blockchain are synchronized with the changes in the external ledger. For this:

1) A structure *(NEXT-BLOCK, hFlag, (txid1, ..., txid1))* is formed based on all transactions in *NEXT_BLOCK_TX*.
2) All outgoing external transactions are deleted from the structure and placed in the *EXTERNAL_TX* variable.
3) For all inbound external transactions, the outbound transaction is deleted from *EXTERNAL_TX*.

Next, the state is configured:

1) The structure *(SET-SLACK, pointers)* is formed based on *SLACK_POINTERS_ROUND*.
2) For those nodes that are not assigned slackness, copy from *SLACK_POINTERS* is performed.

It is also worth noting several important features of this simulator. First, the transfer of transactions to the accepting ledger occurs immediately after the inclusion of an outgoing transaction in the initiating ledger. In a practical implementation, this action can be performed automatically (by notifying the nodes of the target ledger or in an application that creates an inter-system transaction). This action is necessary because additional transaction latency can lead to one of the adverse events that can disrupt the simulation (see below).

Second, there is only one adversary in the model. Although its structure has not been considered before, it is worth noting that this adversary is a wrapper over many nodes that attack each ledger separately. A single node is used to simplify the construction of the model.

Third, this GUC-model assumes that each node maintains only one robust distributed ledger. Due to this simplification, the complexity of the proof is significantly reduced. In addition, nodes that support multiple ledgers are formed on the basis of nodes that support one ledger by combining them using wrapper functionality, similar to creating a multicast functionality based on unicast functionality [9].

The difference between external and internal transactions and the time required for their applying are modeled by temporarily excluding those transactions from blocks created by an ideal robust distributed ledger. This assumption is based on the fact that transactions can be considered complete in the case of scaling only after inclusion in the accepting ledger. During simulation this action can be expressed by placing an intermediate node between nodes that implement a multidimensional blockchain and the environment. It checks the presence of both transactions in the ledgers when requesting information about the status of transactions from the environment.

An alternative approach is the state-slackness setting, when some nodes in the ideal functionality “do not see” the transaction until it is included in the accepting ledger. However, this approach does not allow building a simulation, since new blocks for lagging nodes are not created within a limited time, which means that changes that occur in the corresponding ledgers of the multidimensional blockchain during a round (in the absence of external transactions) cannot be reflected.

Proposition 2. Multidimensional blockchain protocol GUC-implements a robust distributed ledger given 1) condition in Eq. 18 is followed and 2) secure search and verification protocol is used [7].

The simulator presented earlier has been constructed to show that $EXEC_{REAL,A,Z}$ is equivalent to $EXEC_{IDEAL,S,Z}$, since the simulator completely repeats the logic of the adversary’s work by ”translating” messages intended for one model into messages intended for another model. At the same time, as noted earlier, external transactions are ”delayed” until they are included in the accepting ledger.

However, in this case, the simulation is not perfect, as it has not been proven that probability of ”bad” events is negligible. These events are those which lead to the possibility of distinction between the models (from the view of environment Z). Next these events and the likelihood of their occurrence are presented.

BAD1 – violation of the correct functioning of the ledger due to errors in the consensus mechanism. The likelihood of this event is determined by the specific mechanism for reaching consensus and the peculiarities of its implementation. The likelihood of malfunction is negligible, therefore, the following condition is met:

$$p_1 = \prod_{1}^{N} p_i = N * \varepsilon, \tag{14}$$

where N is the number of ledgers inside multidimensional blockchain, ε is negligible variable.

BAD2 – search failure while using search and verification protocol. In this model, an ideal functionality is used, so the probability of this event is taken as negligible. In real world execution, there must be a protocol that GUC-implements the ideal functionality (at least with the restrictions allowed during operation) [7]:

$$p_2 = \varepsilon \tag{15}$$

BAD3 – failed verification while using the search and verification protocol. The probability of this event depends on the verification mechanism. If the approach with the analysis of the initiating blockchain [6] is used, then the probability of this event is 0. If the secure search and verification protocol approach is used [7], then the probability is determined by the initiating ledger chain quality, which is broken with negligible probability:

$$p_3 = \varepsilon, \tag{16}$$

BAD4 – transaction latency for a time longer than window slots, when G_{Ledger} initiates a transaction regardless of the block proposed by the attacker to keep it alive. The probability of this event is determined by the distance between the ledgers,

as well as the delays in requesting information from each ledger (the reciprocal of the connection speed between any two nodes in the network is taken as the worst of all possible options during operation). The distance is calculated with:

$$d = d_1 + d_2, \tag{17}$$

where d_i is the depth of ledger, i.e. number of parent blockchains in the way to the root.

In addition, at least one slot is required to accept an incoming transaction. Therefore, for this probability to be negligible (second condition in statement), it is necessary:

$$window * t_{window} - 1 \geq max\{d_n\} * 2 * max\{1/v\}, \tag{18}$$

where $window$ is the so-called sliding window size [9], i.e. number of rounds during which the state of any two honest nodes is allowed to differ, t_{window} is the size of window round in terms of multidimensional blockchain slots, $max\{d_n\}$ is the maximum depth of any ledger in multidimensional blockchain and v is the connection speed between any two ledgers. Then:

$$p_4 = \varepsilon \tag{19}$$

Consequently, the probability of any of these events occurring when the boundary conditions are met is negligible:

$$P = p_1 + p_2 + p_3 + p_4 = 4 * \epsilon \approx \epsilon \tag{20}$$

Proposition 3. Multidimensional blockchain GUC-implements robust distributed ledger

This proposition is a corollary of Proposition 1 and Proposition 2. Since the multidimensional blockchain protocol GUC-implements a multidimensional blockchain, in the model it can be replaced by the corresponding ideal functionality – according to the universal composition theorem. Hence, the multidimensional blockchain GUC-implements a robust distributed ledger.

V. Conclusion

The paper is devoted to the problem of scaling robust distributed ledgers with the help of multidimensional blockchain.

Thanks to this proof, it becomes possible to use a multidimensional blockchain to build robust distributed ledgers that support scalability, including in automatic mode. At the same time, the most important task remains to build more efficient protocols for the search and verification of blocks and transactions in terms of operating time and volumes of transmitted information.

For such usage to be correct, a two-step proof is constructed. First, using hybrid models, it is shown that multidimensional blockchain protocol GUC-implements corresponding ideal functionality. Then, using simulator, it is shown that this protocol GUC-implements robust distributed ledger.

References

[1] I. M. Shilov and D. A. Zakoldaev, "Multidimensional blockchain security analysis," *Proceedings of Sixth International Congress on Information and Communication Technology. Lecture Notes in Networks and Systems*, vol. 235, pp. 911–924, 2022. [Online]. Available: https://doi.org/10.1007/978-981-16-2377-6_83

[2] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, and E. W. Felten, "Sok: Research perspectives and challenges for bitcoin and cryptocurrencies," *2015 IEEE Symposium on Security and Privacy*, pp. 104–121, 2015. [Online]. Available: https://doi.org/10.1109/SP.2015.14

[3] I. M. Shilov and D. A. Zakoldaev, "Multidimensional blockchain and its advantages," *Informacionnye Technologii*, vol. 4, no. 6, pp. 360–367, 2020. [Online]. Available: https://doi.org/10.17587/it.26.360-367

[4] A. Antonopoulos, *Mastering Bitcoin*. O'Reilly Media, Inc., 2017.

[5] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system." [Online]. Available: https://www.ussc.gov/sites/default/files/pdf/training/annual-national-training-seminar/2018/Emerging_Tech_Bitcoin_Crypto.pdf

[6] P. Gaži, A. Kiayias, and D. Zindros, "Proof-of-stake sidechains," *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 139–156, 2019.

[7] I. M. Shilov and D. A. Zakoldaev, "Security of search and verification protocol in multidimensional blockchain," *Informatics and Automation*, vol. 20, no. 4, pp. 793–819, 2021. [Online]. Available: https://doi.org/10.15622/ia.20.4.2

[8] C. Badertscher, P. Gaži, A. Kiayias, A. Russell, and V. Zikas, "Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability," *CCS '18: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, p. 913–930, 2018. [Online]. Available: https://doi.org/10.1145/3243734.3243848

[9] C. Badertscher, U. Maurer, D. Tschudi, and V. Zikas, "Bitcoin as a transaction ledger: A composable treatment," *Advances in Cryptology – CRYPTO 2017. CRYPTO 2017. Lecture Notes in Computer Science*, vol. 10401, pp. 324–356, 2017. [Online]. Available: https://doi.org/10.1007/978-3-319-63688-7_11

[10] I. M. Shilov and D. A. Zakoldaev, "The robust distributed ledger model for a multidimensional blockchain security analysis," *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, vol. 21, no. 2, pp. 249–255, 2021. [Online]. Available: https://doi.org/10.17586/2226-1494-2021-21-2-249-255

[11] *ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER*. [Online]. Available: https://ethereum.github.io/yellowpaper/paper.pdf

[12] R. Canetti, Y. Dodis, R. Pass, and S. Walfish, "Universally composable security with global setup," *Theory of Cryptography. TCC 2007. Lecture Notes in Computer Science*, vol. 4392, pp. 61–85, 2007. [Online]. Available: https://doi.org/10.1007/978-3-540-70936-7_4

[13] R. Canetti, "Universally composable security," *Journal of the ACM*, vol. 67, no. 5, 2020. [Online]. Available: https://doi.org/10.1145/3402457