Blockchain-Based Task and Information Management in Computational Cloud Systems

Andrzej Wilczyński[1](✉) and Joanna Kołodziej[1,2]

[1] Department of Computer Science, Cracow University of Technology, Warszawska 24, 31-155 Cracow, Poland
{andrzej.wilczynski,joanna.kolodziej}@pk.edu.pl

[2] Naukowa i Akademicka Sieć Komputerowa - Państwowy Instytut Badawczy (NASK-PIB), ul. Kolska 12, 01-045 Warszawa, Poland
joanna.kolodziej@nask.pl

Abstract. Blockchain can be successfully utilised in diverse areas, including the financial sector and the Information and Communication Technology environments, such as computational clouds (CC). While cloud computing optimises the use of resources, it does not (yet) provide an effective solution for the secure hosting scheduling and execution of large computing and data applications and prevention of external attacks.

This chapter briefly reviews the recent blockchain-inspired task scheduling and information processing methods in computational clouds. We pay special attention to security, intrusion detection, and unauthorised manipulation of tasks and information in such systems. As an example, we present the implementation of a new blockchain-based scheduler in the computational cloud. We defined a new Proof of Schedule consensus algorithm, which works with the Stackelberg game, regulates checking and adding new blocks to the blockchain, and determines how to validate schedules stored in transactions. The proposed model assumes competition between different schedule providers. The winner of such a competition takes account of the client's requirements faster and prepares an optimal schedule to meet them. The presented scheduler extends the possibilities of using different scheduling modules by the end-users. By delegating the preparation of the schedules, providers can get benefits only for that, without executing customer tasks.

Keywords: Blockchain · Cloud computing · Security · Scheduling

1 Introduction

The recent digital revolution has contributed to increasing the volume, velocity and variety of data available on the Internet. In the era of Information and Communication Technology (ICT), one struggles with such problems as collecting and optimally managing such large amounts of data. The most important research and challenging engineering tasks related to the ICT systems include ensuring secure network communication of users, data processing and data storage without the nefarious involvement of third parties. The financial aspects are also crucial. One of the possible solutions to such problems may be the application of the Blockchain (BC) technology and networks for the development of the new models of the networks and ICT systems, especially in the cases where security aspects are essential [29].

A rapid progress in developing the new distributed cyber-physical infrastructures of various scales leads to increased demand for computational resources such as computational and data servers, warehouses, databases, networks or services dedicated to data analysis and exploration. The Cloud Computing paradigm has been addressed as a methodology, computing services and architecture to manage these challenges. CC has been defined by Buyya et al. [10] as a simple extension of the grid infrastructure consisting of data centres, where the capabilities of business applications are provided as services that are available in the network. Cloud service providers receive profits for enabling their customers to access such services. On the other hand, consumers are motivated by reducing the related costs. Cloud Computing is not an entirely new model or paradigm but rather an evolution of the following models and technologies:

- **Computational Grid** [19] – a system composed of many connected computers in the distributed clusters [10] that cooperate in a large-scale network.
- **Virtualization of the available resources** – an architectural model, in which many virtual computing devices are visible as one large computing unit. There is no need in the Grid to overhaul the hardware infrastructure to obtain more computing power, and both the infrastructure and computing power are optimally used. Many software tools allow virtualizing machines, for instance: VMware, KVM, Xen [16], or OpenStack Platform [18] which is a more recent and innovative solution. Computational Grid may refer to the hardware resources and the data layer that provides a simpler interface and methods for accessing data. There can be many sources of data, but the user who relies on this data will see one abstract layer [48].
- **Utility Computing Networks** [11] - a model for providing specific resources on-demand and estimating fees based on their consumption;
- **Service-Oriented Architecture** (SOA) [37] - network architectural model focused on the defined services that meet the end users' requirements.

The most popular model of the cloud environment defines Cloud Computing as a multilayer system [39], where the following layer-stack can be specified:

- Infrastructure as a Service (IaaS) - the bottom layer of the system, it provides the client with IT infrastructure such as software, hardware or servicing,
- Platform as a Service (PaaS) - the middle layer, it provides ready-to-use and customized applications without the need to purchase hardware or software licenses,
- Software as a Service (SaaS) - the upper layer, it provides users with specific software features, such as e-mail access or calendar.

The world-leading cloud providers such as Google, Microsoft and Amazon initially used clouds in running their internal business operations. However, after the building of large data centres and data servers farms in many countries [7,8], they noticed a broader potential of the solution. They started offering the external enterprises the previously unused resources or services such as data storage or data processing. There are various methods of classification of cloud environments.

While designing and offering their products to customers, service providers have to take into account the end-to-end personalization of these services and their potential impact on end users' activities, including their business decisions. This is why it is essential to ensure the appropriate service level, which is usually done by concluding a specific contract between providers and consumers, i.e. Service Level Agreement (SLA). As many clients may want to use the same services at the same time, the providers must schedule tasks to achieve the desired quality and pace of service, under the provisions of the SLA. On the other hand, from the providers' perspective, it is essential to minimize the maintenance costs by shutting down resources where unused services are running. To do it quickly, proper scheduling of tasks is necessary.

In practice, the wide-area cloud infrastructure is usually defined as a collection of many cloud clusters working under the same or various cloud standards and administrators. Such distributed cloud clusters are connected by using the standard peer-to-peer (P2P) network. similar model works for the blockchain network, which was the first reason for trying to integrate both environments in order to improve the security policies in global clouds. There are two main methods of integration of the cloud with blockchain platforms:

- Using the cloud for the development of blockchain applications and supporting the integration with enterprise networks (private clouds) to facilitate storage, replication and access to transactional data;
- Using blockchain methods to improve the security of task, user and data management in the clouds.[1]

This chapter briefly reviews the recent blockchain–inspired task scheduling and information processing methods in computational clouds, which are mainly based on the second above mentioned integration method. We pay special attention to security, intrusion detection, and unauthorised manipulation of tasks and information in such systems. As an example, we present the implementation of a novel blockchain–based scheduler in the computational cloud. We defined a new Proof of Schedule consensus algorithm, which works with the Stackelberg game, regulates checking and adding new blocks to the blockchain, and determines how to validate schedules stored in transactions. Such an approach must result in competition between different schedule providers, won by the one who takes account of the client's requirements faster and prepares an optimal schedule to meet them. The presented scheduler extends the possibilities of using

[1] The model of the developed blockchain–based scheduler along with the comprehensive experimental analysis were published in A. Wilczyński's PhD dissertation [49] available at https://doktoraty.iet.agh.edu.pl/_media/2020:awilcz:phd_thesis_aw.pdf.

different scheduling modules by the end-users. By delegating the preparation of the schedules, providers can get benefits only for that, without having to execute customer tasks

The rest of the chapter is organized as follows. In Sect. 2 we present the basic definitions and the concept of the blockchain network. Section 3 presents the recent developments in security-aware scheduling in cloud computing. The new blockchain–based secure cloud scheduler is defined in Sect. 4 along with the examples of the a simple experimental evaluation (see Sect. 4.8). The chapter ends with the conclusions in Sect. 5.

2 Blockchain Backgrounds

There are many definitions of blockchain. Most of them refer to the origins of the blockchain technology that evolved from Bitcoin [36]. Blockchain is usually considered a distributed ledger of records containing cryptographically signed transactions grouped into blocks. The main properties of blockchain technology can be defined as follows [44,55]:

- **decentralization** – in all standard transaction systems, there is typically a central unit called the supervisor, confirming the compliance of the transaction and recording it in the system. Since such a core unit must handle all the requests and approve them, it is often a bottleneck that determines the entire system's efficiency. BC lacks that problematic central supervisor because the decentralized nature of the BC system based on consensus algorithm jointly confirms each transaction, maintaining data consistency,
- **persistence** – since the transactions are validated, any attempt to approve transactions being incompatible with the established policies are immediately detected by confirming/mining nodes; blocks containing incorrect data are immediately detected, too,
- **anonymity** – each user in the network is assigned a generated address (hash) utilizing which they can perform operations. This address does not allow unambiguous identification of a real user,
- **auditability** – each transaction must refer to some previous transactions; hence there is a possibility to trace and verify what has happened with the processed data. For instance, in the bitcoin network, one can check how the balance of a given user has changed since the beginning of its existence in the system,
- **transparency** – transactions of any public address are available for inspection by every user having access to BC; each user of the public network has the same rights,
- **security** – chain of blocks are shared, tamper-proof, and cannot be spoofed due to one-way cryptographic hash functions. The use of cryptographic methods ensures the security of transactions. Roughly speaking, users can send data only if they have a private key. The private key is applied to generate a signature, which in, turn, serves to confirm that transaction was requested by a specific user and to prevent it from being changed,

- **immutability** – data stored in BC are immutable; each entry in the ledger must be confirmed by the network, so it cannot be a secret operation. Each block contains the previous block's hash, which is generated based on the data in the block. Therefore, even a minor change in the data results in the change of the hash and consequent interception and rejection of the modification by the other nodes.

2.1 Blockchain Network

Blockchain architectural model is defined as the distributed network composed of the nodes and users in the following way:

- nodes - individual systems that store the blockchain and ensure transactions are valid;
- users - persons or entities that can read the ledger;
- Peer–to–peer (P2P) – generic architectural model [13], in which each node can communicate with each other without the need of using a central information exchange point.

The abstract model of the BC network is presented in Fig. 1.

There is no general standard in creating a blockchain network that would allow communication between all blockchain networks. For now, each blockchain is made separately, and communications between different blockchain networks require special workarounds. Each BC network works on predefined rules that all nodes in the network agree on. These rules include conditions for adding and validating transactions and the mechanism of interaction between participating nodes. All standard communication rules used by the BC network are called the blockchain protocol.

2.2 Blockchain Components, Protocols and Algorithms

The ledger in the blockchain network is defined as a chain of blocks. Each block contains a hash digest of the previous block. A simple model of the block is presented by Yaga et al. in [54], and it consists of the following components:

1. Block Header:
   - block number;
   - hash of a current block – hash generated from the data contained in the block and previous blocks, usually determined using the Merkle tree (see Fig. 2), any modification of the data in the block will change the hash;
   - hash of a previous block;
   - timestamp;
   - nonce value – the value generated based on *Proof of Work* through solving the hash function, which allows adding the block to the chain.

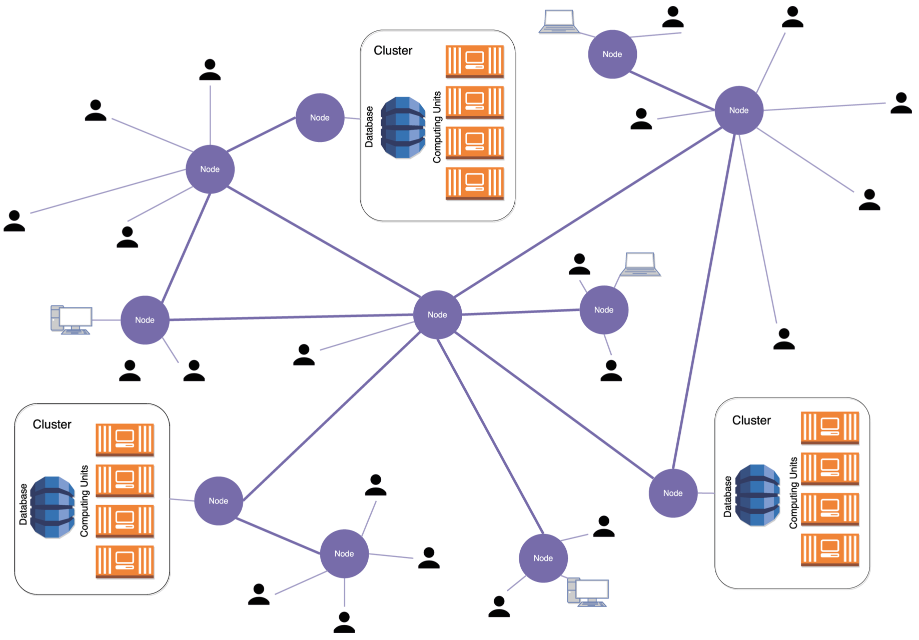

Fig. 1. Decentralized blockchain network Source: [49]

2. Block Data:
   (a) a list of transactions - a single transaction usually consists of:
      – inputs - the input data are usually digital assets to be sent; the source of the asset (its origin) is located here,
      – outputs - in the outputs usually the recipient of digital assets is defined, along with how many digital assets are receiving and conditions that must be met to spend this value,
   (b) other data.

Each ledger starts with a Genesis Block and each following block must be added to the chain after it. The Genesis block defines the initial state of the system. An example of chain of blocks is shown in Fig. 3.

The blockchain operation comprises a few simple steps:

1. The sending node prepares new data as a transaction and broadcasts it over the network.
2. The receiving node verifies the transaction and the data included in it. Each transaction must be signed and authorized using asymmetric cryptography [38]. The private key is used to sign transactions, while the public key is used to identify the user (user address) and verify the signature generated with the private key. This mechanism makes it possible to check whether the user who sent the message is its author. The procedure of signing and verification of the signature is shown in Fig. 4. Whenever the transaction and data pass

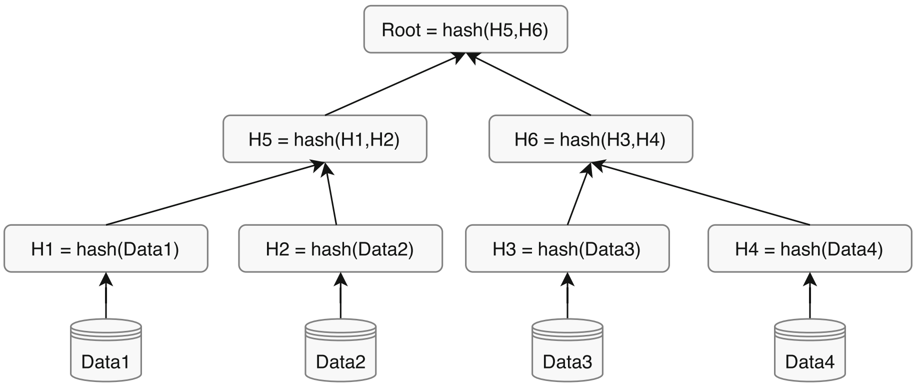

Fig. 2. Merkle tree Source: [49]

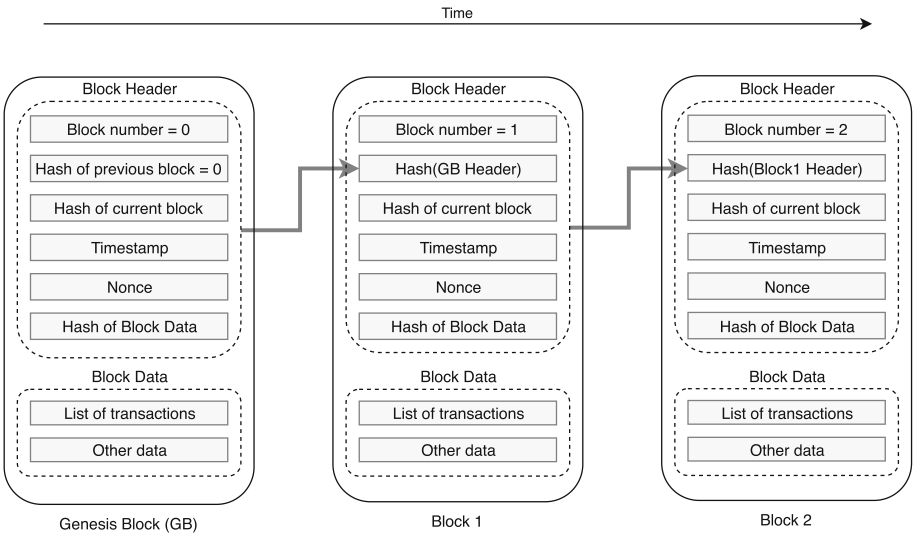

Fig. 3. Generic chain of blocks Source: [49]

the validation process, the node responds to the sending node and saves the transaction in the block.

3. After saving the appropriate number of validated transactions in a local block, nodes start the block confirmation procedure by the consensus adopted in the network.
4. The block is saved in the chain after the execution of the consensus algorithm (block confirmation by the appropriate number of nodes).
5. Every node in the BC network must locally save the approved block and include it in its chain.

It can be noted that many blocks may be published at the same time. Consequently, the existence of different versions of the blockchain in various places

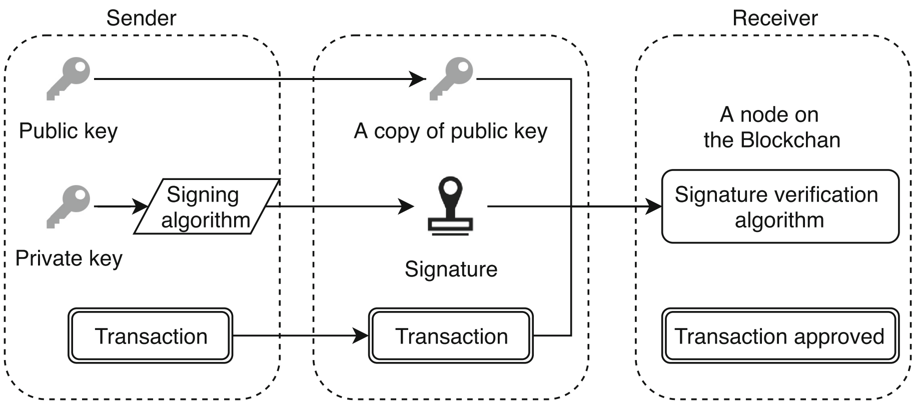

Fig. 4. The procedure of signing and verification of the signature Source: [49]

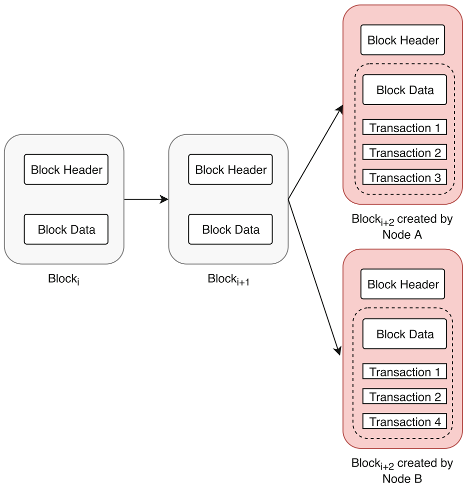

Fig. 5. Blockchain in conflict Source: [49]

is possible. It may happen for various reasons, such as network latency between nodes. This type of problem is shown in Fig. 5. As shown in the figure, a conflict occurs after adding the block $i + 2$. The block added by node A contains transactions 1, 2 and 3 while the block added by node B contains transaction 1, 2 and 4. Therefore, it is not the same block. Such differences do not result from the poor completion of the blocks. They arise from the almost simultaneous confirmation of different transactions by the nodes. Most BC networks deal with this problem while waiting for the addition of the next block by one of the nodes. It is assumed that the *'longer blockchain'* wins. In other words, the winner is the node that is quicker to add the next block after the block, including the above conflict.

2.3 Consensus Models

A key aspect of blockchain technology is determining which node can publish the next block. It requires the implementation of a consensus model. The main techniques for obtaining consensus in blockchain networks include:

– Proof of Work (PoW) - this model is the most popular method of obtaining a consensus inBC network. The node can add a block after solving a computationally intensive puzzle or cryptographic function, which can only be done by brute force [9]. In PoW, the probability of mining a new block by a node depends on the ratio of its computing power intended to solve the puzzle to the total computing power of all miners connected to the network. An example of a puzzle is presented below. The node using the hash function SHA-256 [40] must find a hash value meeting the following criteria:

  SHA256("schedule" + "nonce") = hash value starting with "09"

  The nonce value is a numeric value that is added to the string *'schedule'*. After each hash calculation, the nonce value is changed. This operation is repeated until the hash value has the string beginning with "09".

  SHA256("schedule" + "113") =
  f8ac5c8094a5ebe334ebe4ba1cde6e29acc718743575933
  d6c622406177a6aa4 - means "not solved"

  SHA256("schedule" + "114") =
  a8007fd4ef5eded7a095d958b3af9e89b48b1bc3a313555
  d140f8aa400eb7a6a - means "not solved"

  SHA256(schedule + "115") =
  09ce2827da9ebc62bc2491ce96bdf366044247f17860826
  2154d6eefb8f40721 - means "solved"

  The above puzzle is not complex, but its complexity increases with each addition of a subsequent character to string 09. This model is adopted in such networks where suspicion prevails over the trust. Although it works well, its obvious disadvantage is consuming substantial energy to solve the puzzles. This type of consensus has been used in such networks as Bitcoin or Litecoin (cryptocurrencies).

– Proof of Stake (PoS) - this model combines generating blocks with the possession of a certain amount of digital assets in BC. The selection of a node to perform a function of a validator checking if the next block can be added to the chain is based on the number of assets it includes: the more assets a node has, the more likely it is to be selected. Therefore, the strategy assumes that nodes with more assets can provide more reliable information than those with fewer assets. Usually, tokens are used to determine the number of assets. Assuming that a node in the network has a maximum of 100 tokens and the node has 20 tokens, it has a 20% chance to become a validator and mine a block. Such an approach may lead to a problem related to the monopolisation of the network, where the node with large assets accumulates the assets faster than others. Therefore, in some solutions, limitations associated with adding blocks are applied. After mining the block, the node must wait before confirming the next one. Other solutions introduce limitations of the lifetime of tokens: they are only valid for a specified time. Unlike PoW, PoS does not need to consume much energy to solve puzzles and work more economically. However, nodes have to merge into groups to choose a validator, which causes centralisation. This approach is used, for instance, in Decred [17] or Peercoin [14]. The algorithm proposed by Wilczyński et al. in [49] called *Proof of Schedule* is derived from this consensus model.

– Proof of Authority - in this model, nodes are not asked to solve puzzles or mathematical problems. Instead, the network includes hard-configured units called *'authorities'*, which are authorised to add new blocks and secure the blockchain network. This strategy tends to work well in private or consortium blockchains. Authorities receive a set of private keys with special permissions in the network. The networks based on proof of authority may have some issues concerning the distribution of mining load between signers and the control of the frequency of mining.

2.4 Blockchain Taxonomies

Although blockchain is a relatively new technology, few different blockchain taxonomies are defined in the literature. Lin and Liao [32] divided blockchain technologies into three types, depending on the character of data availability:

– public blockchain - everyone has access to the transaction and can participate in the process of obtaining a consensus; the examples of such a network are bitcoin or Ethereum [46], see Fig. 6,
– private blockchain - not every node can participate in the blockchain network and read the ledger; instead, access is limited and strict access rights management is implemented. A private BC is shown in Fig. 7,
– consortium blockchain - some pre-selected nodes have direct access to BC, and only the nodes from the consortium are allowed to add data; the data to be viewed can be open or private. A consortium BC is shown in Fig. 8.

Another taxonomy is defined for the blockchain network. Cohn et al. in [35] defined the following two classes according to the authorization criterion:

– permissioned blockchains - proprietary networks used by specific persons or entities, for instance, a group of cooperating banks that process financial transactions,

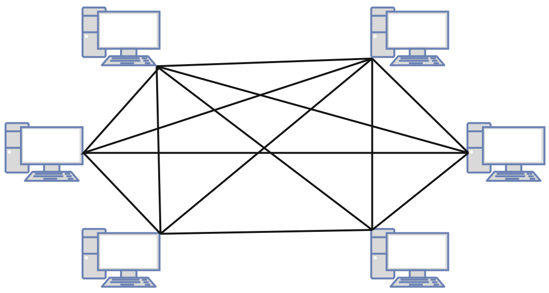

Fig. 6. Public blockchain Source: [49]

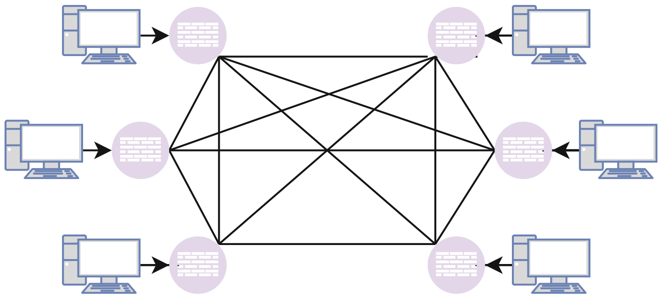

Fig. 7. Private blockchain Source: [49]

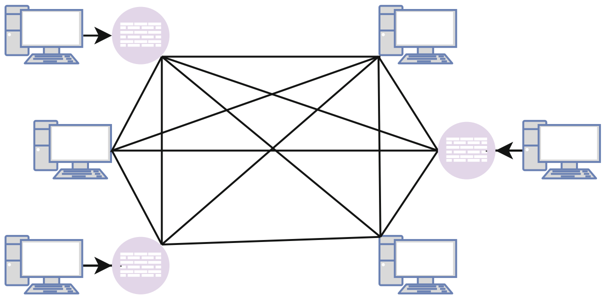

Fig. 8. Consortium blockchain Source: [49]

– permissionless blockchains - open networks to which anyone can access and use data located there.

Considering basic functionality and smart contracts [47] as the main classification criteria, Hileman and Rauches [21] considered two types of blockchain networks:

– stateless blockchain - *'transaction-optimes'*, networks limited to the functionality of the chain in terms of the computational complexity that they can perform (e.g. bitcoin),
– stateful blockchain - *'logic-optimised'*, networks that have expandable functionality in terms of expressing computation (e.g. dApps in ethereum [5]).

Blockchain technology is beneficial, and the particular type of blockchain network should be chosen depending on the service offered, and the market needs.

2.5 Security in BC Networks

Security in BC networks is provided through advanced cryptographic techniques, various methods for determining consensus and, above all, the core aspect of this technology, i.e. immutability of data. Joshi et al. in [24] defined the following important security principles in the blockchain network:

– defence in penetration - the use of multiple layers of security is more effective than the application of a single layer,
– minimum privilege - access to data should be limited to the lowest possible level,
– manage vulnerabilities - security vulnerabilities ought to be constantly checked and corrected,
– manage risks - the risk should be regularly assessed and managed at an environmental level,
– manage patches - faulty system components should be corrected, tested and developed as patches for the successive versions of the application.

Asymmetric-key cryptography is used in blockchain for the authorization of processed transactions. The private key is used here to sign transactions, the public key to identify addresses assigned to the user, and to verify the signatures generated using private keys. Due to asymmetric cryptography, it is possible to determine whether a user who gives a message to another user has a private key with which he has been signed and thus whether he has the right to send it. The transaction is signed with a private key, then together with the signature and public key is forwarded to the recipient. A node in the network based on this information using the verification algorithm can authorize the received transaction.

In practice, consensus models are responsible for providing the appropriate level of BC network safety. For example, in PoW to make the network fake, the node would have to possess at least 51% of the computing resources of the entire network to be able to falsify the information contained in the blocks, which is hardly possible in practice [52].

2.6 Blockchain Usecases

Blockchain technologies have found their way into many practical projects. Some of the most notable projects include:

- **Guardian**[2] – this project defines a token for a new global emergency response network that provides a framework for distributed emergency response systems, especially in locations far from wireless network transmitters and in hard-to-reach areas were calling for help using an emergency number (e.g. 112 or 911) is virtually impossible.
- **Blockchain Charity Foundation**[3] – it is a non-profit foundation whose mission is to create a decentralized network of charitable foundation centres, promote sustainable development, and ensure that they are as widely accessible as possible to those most in need.
- **Power Ledger**[4] - a system that allows customers to choose a source of electricity, enabling electricity trading with their neighbours and ensuring a fair return on investment, where energy is stable and affordable for everyone.
- **EthicHub**[5] - a system whose aim is to make available to all clients also individual investors the same access to traditional financial services by democratizing finances and making available investment opportunities around the world.
- **Grassroots Economics & Bancor**[6] – the decentralized blockchain-based community currencies in Kenya, aimed to combat poverty by encouraging local and regional trade.
- **VeChain**[7] – decentralized platform in which companies can quickly establish contacts and make transactions without intermediation.

The above examples show that the use of this technology ensures high security and quick execution of transactions and enables solving problems that have not been solved in any other way.

3 Security Criterion in Scheduling and Resource Allocation Problems in Computational Clouds

Task scheduling aims to build a schedule that determines when to execute each task and which resources should be selected to do it. For instance, tasks must be scheduled when there is a need to perform several calculations provided by the users and deliver the results within a specific time. To ensure a guaranteed Quality of Service (QoS) [27] to the clients, it is necessary to make as efficient

[2] https://guardtoken.net.
[3] https://www.binance.charity/.
[4] https://www.powerledger.io.
[5] www.ethichub.com.
[6] www.businesswire.com/news/home/20180621005727/en/Bancor-Launch-Blockchain-Based-Community-Currencies-Kenya.
[7] www.investinblockchain.com.

mapping of tasks to the given resources as possible; otherwise, the clients will not pay for them. Therefore, task scheduling is considered one of the burning issues to tackle in cloud computing systems.

A generic task scheduling model is shown in Fig. 9. In the figure, clients directing requests to the cloud can be seen. The cloud broker (task scheduler) collects the requests, responsible for decomposing requests for smaller tasks and running them to virtual machines. After the tasks are executed, the results are returned to the broker, passing them to the cloud client.

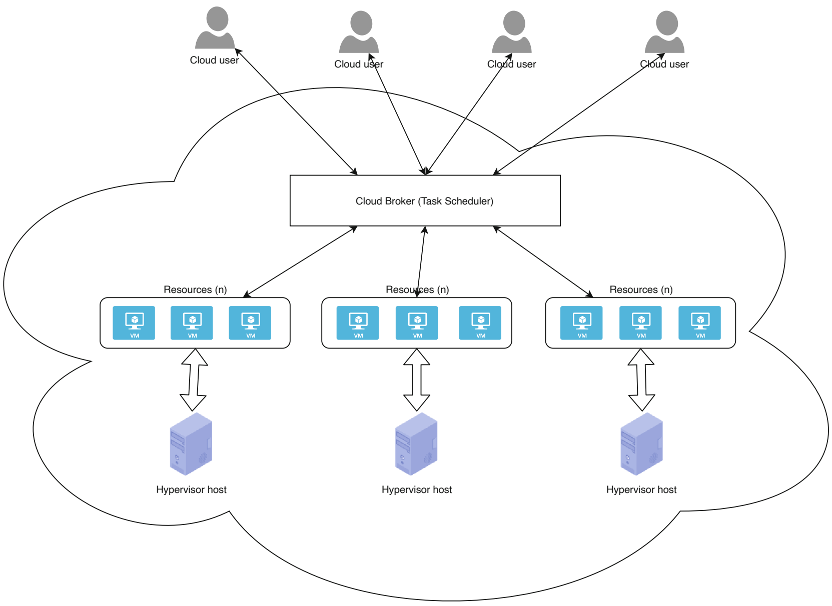

Fig. 9. Basic task scheduling process in cloud computing Source: [34]

Task scheduling in computational clouds can be considered, in fact, as a set of problems. Specification of the problem instances may be formulated based on various cloud scheduling attributes, namely:

– the environment (static or dynamic),
– cloud architecture (centralized, decentralized or hierarchical),
– task processing policy (immediate or batch),
– tasks' interrelations (independency or dependency).

Karatza et al. [20] defined the following instances of the cloud scheduling problems:

– bag-of-tasks scheduling - jobs consisting of independent tasks that can be processed in parallel,

- gang scheduling - jobs consisting of tasks that often communicate with one another, which can be processed in parallel,
- Directed Acyclic Graph (DAG) scheduling [45] - jobs consisting of tasks with a significant order of execution (workflow); tasks can be planned on different system nodes,
- real-time scheduling - composed of jobs in which the deadlines for executing tasks are defined,
- fault-tolerant scheduling - jobs in which there is a high probability of software failures that may prevent the execution of the schedule.

Most of the problem instances defined above are multi-objective global optimization problems. Such objectives, defined as the optimization criteria, are usually minimized to execute the generated schedules faster and cheaper for the clients or to do the individual tasks at a predetermined time. The most popular optimization criteria are the following:

1. Makespan – the time of finishing the last task from the batch; the smaller the makespan is, the faster the tasks are completed:

$$makespan = max\{ET_i, ET_{i+1}, ..., ET_n\} \tag{1}$$

   where
   ET_i the ending time of the task i
   n number of tasks in the batch
2. Flowtime - the sum of the ending times of all tasks from the batch; this metric describes the response time to the client for the submitted task, and its minimization means a reduction in the average response time of the entire schedule:

$$flowtime = \sum_{i=1}^{n} ET_i \tag{2}$$

   where
   ET_i the ending time of the task i
   n number of tasks in the batch
3. Economic cost - the total sum the client has to pay to the provider for the resource utilization

$$Economic\ Cost = \sum_{i=1}^{m} (C_i * T_i) \tag{3}$$

   where
   C_i the cost of 1 second of utilization the resource i
   T_i time in which the resource i is utilized
   m number of resources

4. Resource utilization - maximizing the utilization of the resources, this metric is very important for the provider whose profit raises with the reduction of time gaps when the machine is not utilized:

$$Resource\ Utilization = \frac{\sum_{i=1}^{m} \ TR_i}{makespan * m} \tag{4}$$

where
TR_i the time of completion of all the tasks by the resource i
m number of resources
5. Deadline constraint - defines the time limit within which the task or batch must be executed.

More cloud scheduling criteria, such as tardiness, waiting time, turnaround time, fairness, throughput, priority constraint, dependency constraint, budget constraints, are defined in [25].

3.1 Security-Aware Cloud Schedulers – A Short Survey of Recent Schedulers

While the maximization of the resource utilization and profits of the resource owners are the key objectives of cloud scheduling, they may conflict with cloud users' security requirements and system reliability. Network security threats cause a significant hurdle to ineffective job and service outsourcing in the cloud. The cloud cluster or the cloud resource may not be accessible to the scheduler when infected with intrusions or malicious attacks. In such cases, the failures of the cloud resources and services during the tasks' executions can be observed.

The security-related scheduling criteria cannot be defined as the optimization ones. They are usually connected with monitoring all cloud users' system performance, decision strategies (including end-users), secure data, and information storage and processing.

In this section, we briefly survey the most influential models of secure cloud schedulers and demonstrate the usefulness of the blockchain methods.

In the first analyzed model, Kołodziej and Xhafa [30] proposed a scheduling model simultaneously allowing aggregation of task abortion and ensuring security requirements, which are the criteria for the cumulative objective function along with makespan and flowtime. They defined a meta-broker as being responsible for checking the security conditions and availability of resources. The level of security in their approach is determined based on trust level (tl) parameters defined for the resources and security demand (sd) defined for the tasks. These parameters depend mainly on the user's specific requirements, security policy, history of attacks, or the ability to self-defence. They are described in more detail in [28]:

- security demand - related to tasks, specified for each task in the job, refers to data integration, task sensitivity, peer authentication, access control and task execution environment, is defined as a vector:

$$SD = [sd_j, sd_{j+1}, \ldots, sd_n] \tag{5}$$

where

sd_j one of security demand parameters, assumes a value within the range [0,1], where 0 represents the lowest and 1 the highest security requirements for execution task j

n number of tasks in the job

- trust level - related to resources, specified for all resources in the system, this metric determines the level of client trust to the resource manager, refers to prior task execution success rate, cumulative grid cluster utilization, firewall capabilities, intrusion detection capabilities, intrusion response capabilities, is defined as a vector:

$$TL = [tl_i, tl_{i+1}, \ldots, tl_m] \tag{6}$$

where

tl_i one of trust level parameters assumes a value within the range [0,1], where 0 represents the riskiest and 1 fully trusted machine i

m number of resources

Having SD and TL vectors, it is possible to assess whether the condition of ensuring security is met and, consequently, whether the task can be successfully executed on a given machine. It means that $sd_j \leq tl_i$ for a given (j, i) task-machine pair. In the experimental section, the authors compared the results of scheduling carried out in 2 different modes: a secure mode where all the security conditions and resource uncertainty are verified for the task-machine pairs and a risky mode where all risky and failing conditions are ignored. The measurement of the makespan showed that, in comparison to the classic approach, some scheduling algorithms performed better in risky mode when put in Grid environments having medium or large size. On the other hand, the secure mode brought the best results in all grid instances. The referred article addresses security issues but its scope is fairly narrow and theoretical. It does not discuss such issues as checking the inviolability of tasks and results, unauthorized modification or correctness of the prepared schedule.

Another model was developed by Li et al. in [31]. They defined a model of the security and cost-aware scheduler (SCAS) for different types of tasks in computational clouds, intended to minimize the total cost of workflow execution while meeting the assumed deadline and risk rate limits. Their approach was based on the application of Particle Swarm Optimization algorithm (PSO) to create a workflow schedule with tasks mapped to the resources and to the type and number of virtual machines that should be used. To protect the tasks against snooping, alteration and spoofing attacks, the authors used three security services: authentication, integrity, and confidentiality. Each task can require all three security measures, with the security levels depending on the user's specification. In the experimental section, four algorithms were tested against three workflows. Then, the impact of security services and risk coefficient were examined. The results confirmed the effectiveness and practicality of the used algorithm.

Jakóbik et al. in [23] present an innovative architectural model based on a multi–agent scheme and security-aware meta scheduler controlled by genetic heuristics. The authors focused on the safety of task scheduling in cloud computing and described its behaviour in the event of a task injection attack. Namely, they considered a situation in which an attacker, logged in as an authorized consumer, tries to send an unauthorized task (see Fig. 10). It triggers a response from the system in the form of an alert sent to the correct place: its verification takes place before task scheduling. In addition, the authors proposed two models supporting users security requirements, a scoring model that allows task scheduling only on virtual machines that have an appropriate level of security and a model that considers the time needed for cryptographic operations associated with each specific task. These models are similar to those described by Kołodziej, and Xhafa [30]. In the experimental part, the influence of non-deterministic time intervals for the scheduling process on the environment performance was examined, and the makespan for different security levels was calculated. The results showed the effectiveness of the proposed models and their increasingly positive impact on the system's safety.

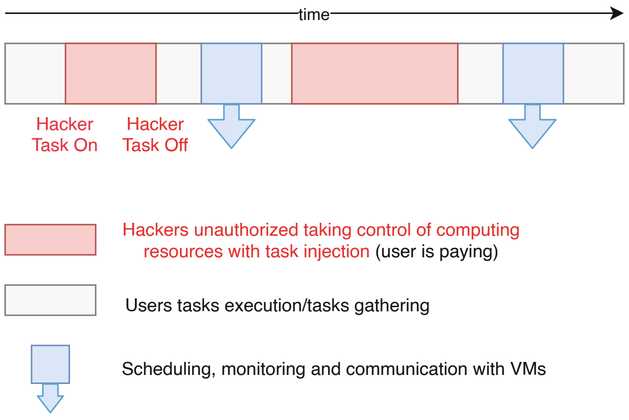
time
Hacker Task On
Hacker Task Off
Hackers unauthorized taking control of computing resources with task injection (user is paying)
Users tasks execution/tasks gathering
Scheduling, monitoring and communication with VMs

Fig. 10. Task injection attack proposed by Jakóbik Source: [23]

On the other hand, one should mention a relatively recent study by Lokhandwala [33] which is mainly related to the topic of this paper because its author resorted to blockchain technology to solve the problem of task scheduling. In Lokhandwala's approach, a decentralized blockchain network was used to allocate resources more efficiently, resulting in reduced energy consumption and, consequently, costs. A load of data centres stored in blocks is checked using smart contracts [47]. Then, the tasks to be executed are assigned to the data centres with the most negligible load. The algorithm on which the smart contract was based is shown in Fig. 11. In the experimental part, the correctness of the blockchain network was first checked and, subsequently, the model was evaluated. To conduct experiments *Shortest Job First* (SJF) algorithm was applied

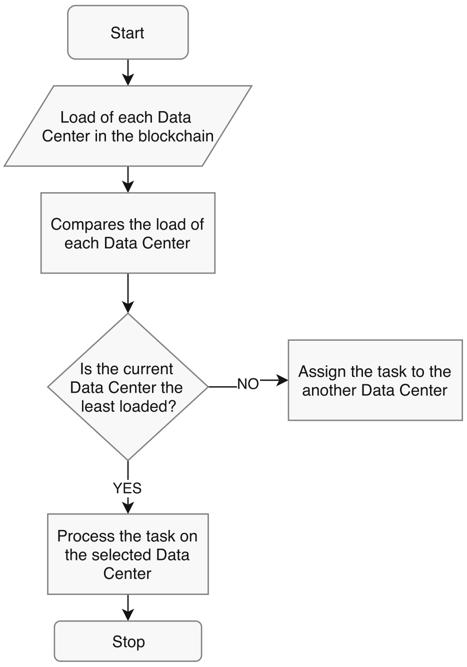
Start
Load of each Data Center in the blockchain
Compares the load of each Data Center
Is the current Data Center the least loaded?
NO
Assign the task to the another Data Center
YES
Process the task on the selected Data Center
Stop

Fig. 11. Smart contract algorithm proposed by Lokhandwala Source: [33]

whose primary purpose was to minimize the waiting time of virtual machine (VM) response. However, the author did not measure the actual impact of the method on the waiting time of VM response, which would require its comparison with one of the classical methods not based on blockchain. The focus was more on testing the functioning of the blockchain network, which included assigning the tasks to the appropriate data centres and the security issues, i.e. blockchain resistance to manipulate the data. Lokhandwala concluded that blockchain was more suitable for data storage than calculating the load of Data Centres. The block mining process turned out to be very energy-consuming due to the chosen consensus algorithm (which probably should have been different for the case).

Hong et al. [22] discussed the problem of communication and task scheduling among users in device-to-device network (D2D) [26] to reduce the average time of task execution effectively. Their idea consisted of the wasted computing power of mobile devices, which are typically in an idle state with nothing but notification listeners and other low energy consumption applications activated. The possibility of using these static resources together with the storage can lead to highly profitable and profitable cooperation in executing tasks in D2D networks. There are, however, some doubts if task scheduling in such systems is fair to everyone. It may look unfair if the users contributing a lot of their computational resources to others receive little being in dire need. Hence, Hong et al. proposed an innovative blockchain-based credit system that can be used for task

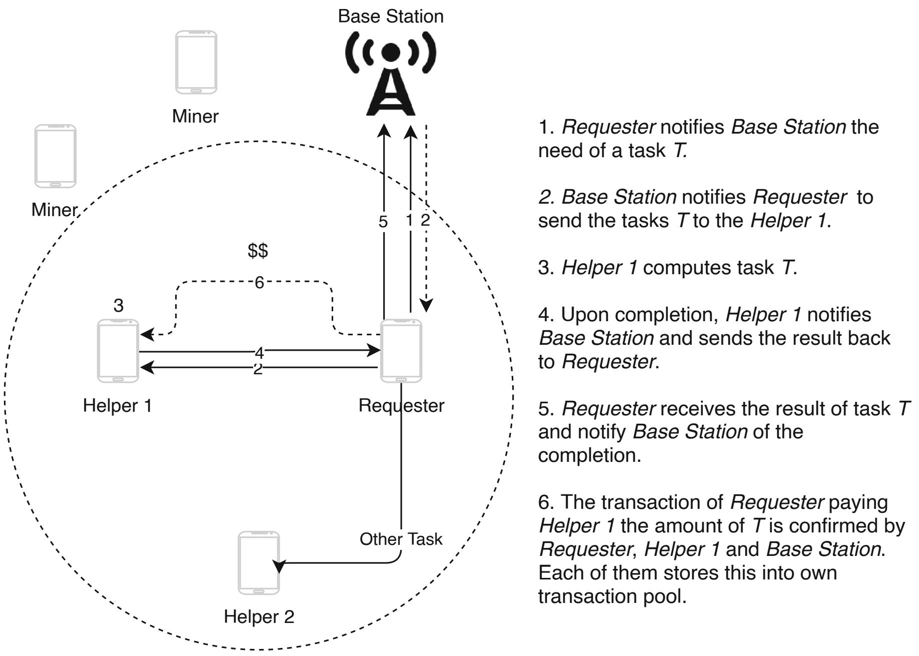
Base Station
Miner
Miner
$$
6
3
4
2
5
1
2
Helper 1
Requester
Other Task
Helper 2
1. *Requester* notifies *Base Station* the need of a task *T*.
2. *Base Station* notifies *Requester* to send the tasks *T* to the *Helper 1*.
3. *Helper 1* computes task *T*.
4. Upon completion, *Helper 1* notifies *Base Station* and sends the result back to *Requester*.
5. *Requester* receives the result of task *T* and notify *Base Station* of the completion.
6. The transaction of *Requester* paying *Helper 1* the amount of *T* is confirmed by *Requester*, *Helper 1* and *Base Station*. Each of them stores this into own transaction pool.

Fig. 12. Task scheduling in D2D network proposed by Hong Source: [22]

scheduling to enforce justice among D2D network users. Their solution consists of cooperative task scheduling to reduce the average task execution time among the users and a blockchain-based credit system to ensure fairness in the network. The system model and the principle of its operation are presented in Fig. 12. The authors checked the impact of various initial credits provided to each user, different maximum waiting times, task sizes, and time elapsed on the performance. According to the results, the proposed model significantly shortens the average task execution time for the requesters in D2D networks.

4 Blockchain-Based Secure Cloud Scheduler

Task scheduling systems available on the market (e.g., Amazon ECS[8]) use many variations of scheduling algorithms (FCFS [53], SJF [12] etc.) that are, nevertheless, not publicly available. Cloud service providers such as Amazon Web Services (AWS)[9], or Azure[10] use their algorithms to prepare schedules. The clients wishing to use their services send the task to them and can only trust that it will be done according to their expectations at the lowest possible cost. The internal

[8] https://docs.aws.amazon.com/AmazonECS/latest/developerguide/Welcome.html.
[9] https://aws.amazon.com/.
[10] https://azure.microsoft.com/pl-pl/.

algorithms used by service providers are not always optimal in taking account of the specific customer requirements.

In this section, a new *Secure Blockchain Scheduler* (BS) model defined in [50] was described. In this model, the specific customer needs and security requirements are considered. We implemented the public blockchain to make the model available to every scheduler provider who would like to participate in preparing the schedule. The provider who prepares the schedule meeting the client's requirements faster wins. In this case, the inter-clouds [41] approach is referred to, in which the cooperation between cloud providers is possible whenever they need additional services or computing resources. The defined model is based on a similar idea according to which the providers can use the services offered by other providers to obtain the best schedule. Communication between different cloud providers is determined by a public blockchain, which is decentralized and does not require the use of unique protocols for information exchange. Therefore, the units specializing only in one field and not necessarily providing other services can freely participate in the generation of schedules. The general concept of the developed model is presented in Fig. 13. The main actors in that model are clients (end users) and cloud service providers. Its main elements are the pool of requests, transactions, nodes participating in the transaction approval process and chain of blocks in which transactions with prepared and confirmed schedules are located and stored.

4.1 Clients and Cloud Service Providers

Clients formulate their requirements and direct them to the cloud in our scheduling model. These can be jobs related to running the executable file or executing a source code fragment that solves some mathematical problem. Together with a specification of their needs, they can also provide some specific conditions, for instance:

- executing tasks on a machine with special security parameters (firewall, antivirus, etc.) due to data sensitivity and confidentiality,
- data processing only on servers located in a defined geographical location due to legal reasons,
- quick execution of tasks regardless of costs or the exact opposite.

Cloud service providers collect requests from clients, which are then forwarded to *Task Managers* (TMs). Based on the received data describing the tasks and the specific requirements of the client, TM who knows the available resources/virtual machines (VMs) in his cloud, chooses the ones that will be the most suitable for their execution. The choice is made considering requirements in terms of security, the physical location of machines, the short waiting time for results, or the client's small budget. After that, the task manager performs a description of the selected VMs and tasks to be executed.

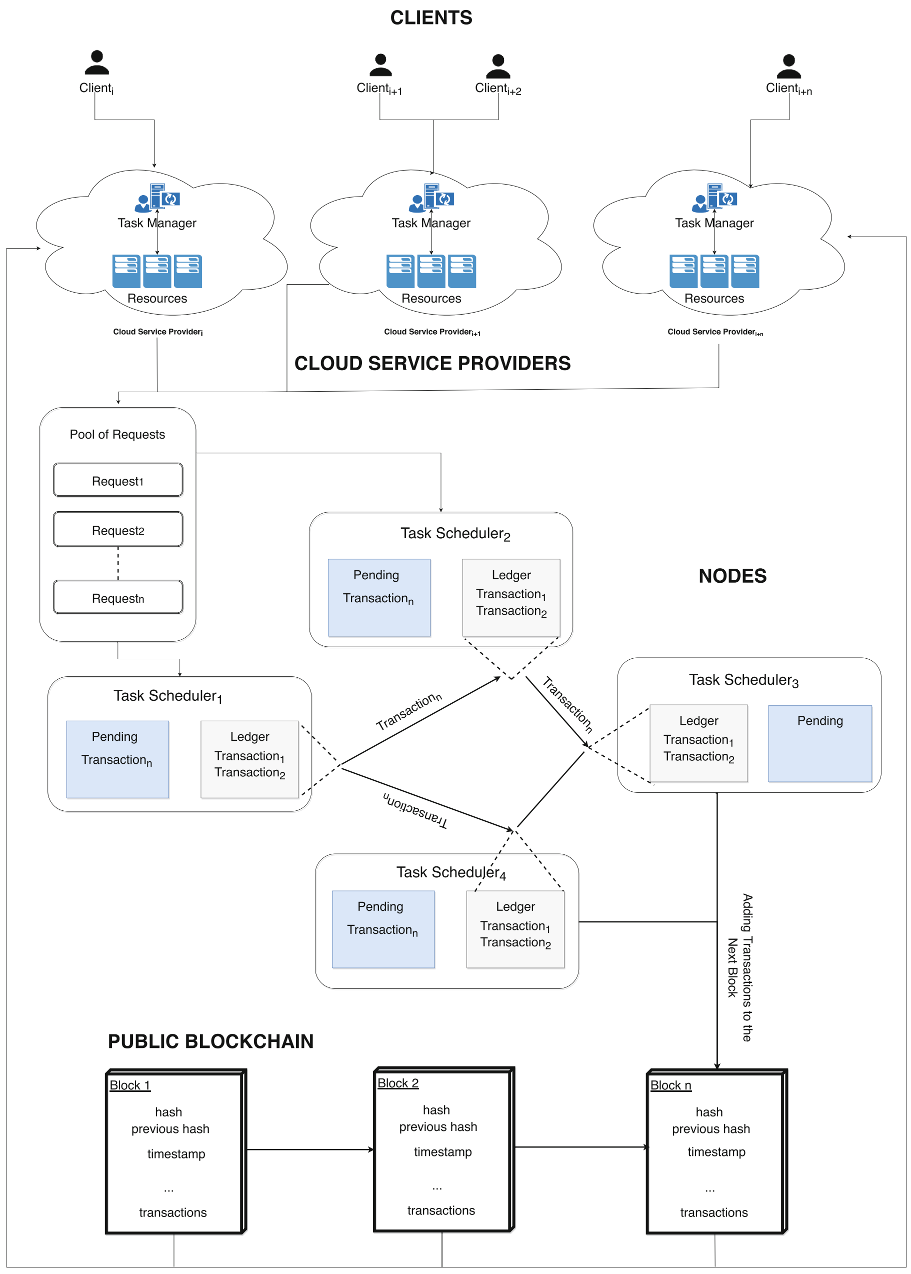

Fig. 13. Secure blockchain scheduler model Source: [49]

Each VM is characterized using two factors:

- computing capacity cc,
- trust level tl (Eq. 6).

Each task is characterized using two factors:

- workload wl,
- security demand sd (Eq. 5).

In addition to the specification of machines and tasks, mainly based on security aspects specified by the client, TM also sets the value of the expected SL parameter for the schedule to be prepared. If the SL value is equal to 0 then all proposed schedules designed by the nodes are accepted; otherwise, only those whose SL value is greater than or equal to the value given by TM are considered as correct. The SL parameter assumes values in the range [0, 3], and the higher its value, the longer the schedule preparation because the majority of prepared schedules are rejected. On the other hand, the higher the SL value, the more secure the schedule is.

4.2 Pool of Requests

After preparing the characteristics of the tasks and machines and defining the expected SL, all the information is defined as the request, which is then sent and collected in the request pool. At this stage, the task manager also signs the request, i.e. the future recipient of the target transaction containing the prepared schedule. A body of such request is shown in Fig. 14. The requested pool is generally available, and task managers from different clouds can bid. Nodes located in the blockchain network select those requests they would like to generate the schedule.

4.3 Nodes and Transactions

The initiating node retrieves the request from the request pool and prepares the schedule for the tasks and machines described according to its scheduling algorithm. After the preparation, it calculates one of the scheduling criteria, i.e. makespan or flowtime or economic cost or resource utilization. The obtained results are placed into the transaction with data from the request and then broadcast over the blockchain network for confirmation. The fully prepared transaction is shown in Fig. 15 and contains such information as:

- id - transaction id generated based on data contained therein,
- sender (PKN) - the public key of the node preparing the transaction,
- recipient (PKTM) - the public key of the task manager creating the request,
- signature (DS) - digital signature made by the node,
- request id - id of the request sent by the task manager,
- information about the prepared schedule:

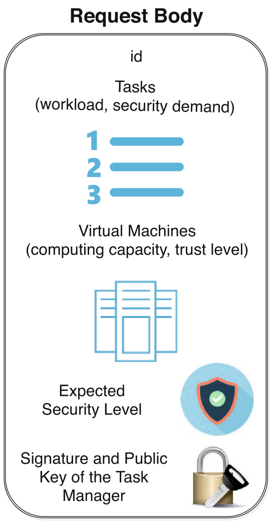

Fig. 14. Body of request from the pool of requests Source: [49]

- tasks - id, workload and security demand of the tasks;
- machines - id, computing capacity, trust level and ids of tasks to execute; each machine includes the assigned ids of tasks that should be executed on it;
- scheduling criterion (SC) - makespan, flowtime, economic cost or resource utilization.

In the defined transaction, SC is the factor that determines whether the schedule is optimal. This factor will be replaced by makespan, flowtime, economic cost or resource utilization. In the experimental part, the model will be evaluated for each criterion. After obtaining the appropriate number of transaction confirmations, the node places it in the block. Before adding the transaction to the block, the SL value of the prepared schedule is calculated. If the value of SL is, at least, at the security level specified by TM, the transaction is added to the block. Otherwise, it is omitted because further processing and saving in the chain of blocks would be pointless.

4.4 Chain of Blocks

The block is created and validated after collecting a sufficient number of transactions, and the number of required transaction confirmations is defined as a global parameter for the entire BC network. Each block consists of:

- the block hash value (Bhv) - calculated on the basis of previous block hash value, a timestamp and the Merkle tree root hash value,
- the previous block hash value (PBhv),

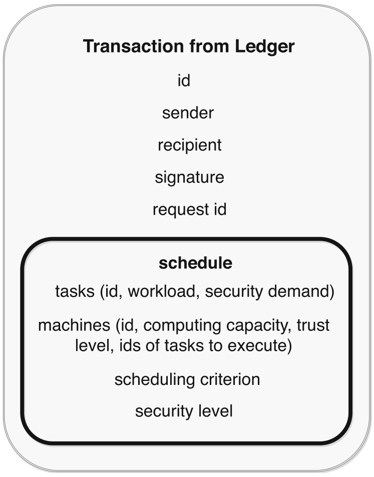
Transaction from Ledger
id
sender
recipient
signature
request id
schedule
tasks (id, workload, security demand)
machines (id, computing capacity, trust level, ids of tasks to execute)
scheduling criterion
security level

Fig. 15. Body of the transaction Source: [49]

– a timestamp (Tim),
– the Merkle tree root hash value (MTRhv) (see Fig. 2), generated using the SHA-256 hash function [40],
– a list of transactions - transactions with prepared schedules, containing the information presented in the Fig. 15.

Once all the block building requirements are met, it is mined by mining node and distributed across the BC network. During block propagation, some conflicts may occur because several nodes may try to add many blocks simultaneously. As a result, different versions of the chain of blocks may be provided (see Fig. 5). Such situations are resolved by adopting a rule to wait for the next added block and invariably recognize a more extended sequence of blocks as official; in other words, the first node successful in adding the following block prevails. After confirming the chain throughout the BC network and recognizing it as official (a fragment of such a chain is shown in Fig. 16) task managers can load the schedule prepared for them. They obtain it using their public key; each transaction with a prepared schedule is addressed to the TM whose PKTM was given in the pool of requests. Having the schedule, TM can allocate tasks to the resources and monitor them. At this moment, the whole process ends.

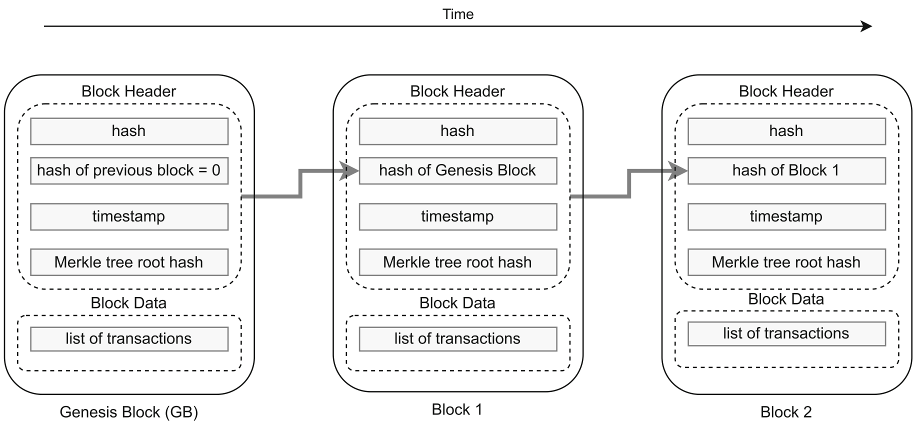

Fig. 16. The first few blocks from the chain of blocks of the model Source: [49]

4.5 Proof of Schedule – Generalized Stackelberg Game

We introduce in our model a new consensus algorithm – *Proof of Schedule* (PoSch) [50], based on the Stackelberg game. Our consensus algorithm is dedicated to regulating schedule checking and adding new blocks to the chain.

Let us denote by i to l the nodes (task schedulers), which participate in the approval of the schedule (transaction). Node ts_i, based on the pool of tasks and their workloads and the collection of virtual machines and their computing capacities, prepares the most optimal schedule according to its algorithm and gives its adopted scheduling criterion; let's assume that in this case, it is makespan. After the preparation, the schedule is placed in the transaction and disseminated across the network. The nodes that receive the transaction for confirmation also prepare the schedule according to their algorithms and calculate its makespan. The following notation will be used to present the described procedure:

- $TS = \{ts_i, ts_{i+1}, \ldots, ts_l\}$ - the set of nodes involved in confirming transaction,
- wl_j - characterization of the task j,
- $wl(schedule)$ - the sum of vector elements $[wl_1, \ldots, wl_n]$, defined for all tasks from schedule,
- ts_i - node initiating the transaction,
- ts_{i+1} - first node confirming the transaction,
- M_{ts_i} - makespan of the schedule determined by node ts_i,
- $M_{ts_{i+1}}$ - makespan of the schedule determined by node ts_{i+1},
- SF_{ts_i} - scheduling factor of the ts_i node, the sum of $wl(schedule)$ for all schedules in the blockchain added by node ts_i,
- $SF_{ts_{i+1}}$ - scheduling factor of the ts_{i+1} node, the sum of wl_{sch} for all schedules in the blockchain added by node ts_{i+1},
- BW_t - the sum of $wl(schedule)$ for all schedules added to the blockchain within a given period of time t.

Each subsequent node verifies the schedule sent by its predecessor, which does not mean that the predecessor prepared it (it can be a schedule from the initiating node). First, it prepares the schedule according to its algorithm and then calculates its makespan. The schedule described above confirmation/approval can be formally modelled using Generalized Stackelberg Game model. In this game, there are many players and the game is defined as sequential with the choice of strategy by one player in each stage. In first stage, the game takes place between nodes $ts_i - leader$ and $ts_{i+1} - follower$; the assumption is that the node ts_i has already made its move. The next move is performed by node ts_{i+1}, which has two options to choose:

- M_{ts_i} - makespan proposed by the *leader* ts_i;
- $M_{ts_{i+1}}$ - makespan calculated by its own algorithm.

The $follower$ chooses one of two pure strategies defined as:

- s_1 - choosing makespan M_{ts_i};
- s_2 - choosing makespan $M_{ts_{i+1}}$;

where s_1, $s_2 \in \{0, 1\}$.

To determine the utility function for the follower, the scheduling factors, which are treat as confidence coefficients must be scaled, which is carried out as follows:

$$\overline{SF_{ts_i}} = \begin{cases} 1 & \text{if } \max\{SF_{ts_{i+1}}, SF_{ts_i}\} = SF_{ts_i} \\ \frac{SF_{ts_i}}{SF_{ts_{i+1}}} & \text{if } \max\{SF_{ts_{i+1}}, SF_{ts_i}\} \neq SF_{ts_i} \end{cases}$$
$$\overline{SF_{ts_{i+1}}} = \begin{cases} 1 & \text{if } \max\{SF_{ts_i}, SF_{ts_{i+1}}\} = SF_{ts_{i+1}} \\ \frac{SF_{ts_{i+1}}}{SF_{ts_i}} & \text{if } \max\{SF_{ts_i}, SF_{ts_{i+1}}\} \neq SF_{ts_{i+1}} \end{cases} \tag{7}$$

The utility function for the $follower$ depends on both strategies s_1 and s_2 and scaled confidence coefficients of players:

$$u(s_1, s_2, M_{ts_i}, M_{ts_{i+1}}) = M_{ts_i}\overline{SF_{ts_i}}s_1 + M_{ts_{i+1}}\overline{SF_{ts_{i+1}}}s_2 \tag{8}$$

Looking for a strategy is tantamount to solving the following problem:

$$\begin{cases} \underset{s_1, s_2}{\text{argmax}}\ u(s_1, s_2, M_{ts_i}, M_{ts_{i+1}}) \\ s_1 + s_2 = 1 \\ s_1, s_2 \in \{0, 1\} \end{cases} \tag{9}$$

The problem 9 is a maximization problem with constraints, which was solved using the simplex method [42]. Once the solution, i. e. strategies s_1 and s_2, are found, node ts_{i+1} chooses:

– the option M_{ts_i}, if $s_1 = 1$ and $s_2 = 0$,
– the option $M_{ts_{i+1}}$, if $s_2 = 1$ and $s_1 = 0$.

If node ts_{i+1} loses the game (it chose the option M_{ts_i}), it sends transaction from node ts_i to the next node ts_{i+2} and notifies the node ts_i about the correctness of its schedule, so the next stage of the sequential game is taking place, where node ts_i remains the leader and the follower is the next node ts_{i+2}. Otherwise, node ts_{i+1} initiates a new transaction according to its own schedule, becoming a *leader* and sends it for verification to the node ts_{i+2}, which is the first $follower$. The game is carried out until the node receives confirmation from the appropriate number of TS sequence items. The fewer confirmations the system requires, the more transactions it can produce in a given period. On the other hand, the more confirmations, the more secure and reliable the system is in comparison to the other systems [6]. This value should be selected individually, depending on the system implementation. In the proposed approach, the minimum number of transaction confirmations (MTC) is defined as a global parameter during blockchain initialization. SF_{ts_i}, $SF_{ts_{i+1}}$ coefficients must be non-zero, so if nodes participating in the game do not yet have such data, they must be randomly selected. Similarly, the initiating node ts_i, having no predecessor, will choose its time of schedule.

4.6 Blocks Mining

The block can contain many transactions. It is ready to be confirmed if the sum of $wl(schedule)$ for all schedules within all block transactions exceeds the set value:

$$\text{block is ready to mine if} \quad \sum_{i=1}^{p} \; wl(schedule_i) \geq B_{wl} \tag{10}$$

where

- $wl(schedule)$
- B_{wl} minimum block workload defined as a global parameter set during blockchain initialization
- p number of transactions in the block

When the appropriate B_{wl} indicator is obtained, the mining node can add a block to the blockchain. Each transaction in the block must be validated. Validation is intended to check whether the providers of the transaction inputs have cryptographically signed the transaction. The signature verifies if they have the right to transfer funds for participation in preparing the schedule. Confirmation nodes check the published block, demonstrating that each transaction it contains has been validated, after which the block can be added to the blockchain. The subsequent blocks are added by nodes called *validators*. $Validators$ are nodes that have so far participated in the transaction creation process and have also expressed a desire to mine a block. One leader is selected from the pool of validators. The election is based on adding a node's transaction history covering a specified period. The selection criterion is defined as:

$$\begin{cases} L_t = max(TF_{(v_i,t)}, TF_{(v_{i+1},t)}, ..., TF_{(v_{i+n},t)}) \\ TF_{(v_i,t)}, TF_{(v_{i+1},t)}, ..., TF_{(v_{i+n},t)} \leq \frac{1}{2}BW_t \end{cases} \tag{11}$$

where

- L_t - a leader for adding the given block in the given time t
- $TF_{(v_i,t)}$ - trust factor - the sum of all I added to the blockchain by validator v_i during time t
- BW_t - the sum of all I added to the blockchain during time t

The above approach originates from the idea of *Proof of Stake*, namely from its specific case referring to the hybrid of coin ageing systems and delegate systems [54]. A node can become a leader only if its current trust factor does not exceed the value of $\frac{1}{2}BW_t$, which protects BC network against the 51% attack (or majority attack) to which such systems are vulnerable [1].

4.7 Profits for Task Schedulers

Nodes involved in creating or confirming transactions and block mining can be rewarded in various ways. They usually charge a fee for performing specific actions. One of the possibilities is presented below. According to it, nodes (task schedulers) can receive profits for participating in operations on which the functioning of BC network is based. The task scheduler profit for creating or confirming one transaction (schedule) $P(schedule)$ within an entire block can be defined as follows:

$$P(schedule) = \frac{wl(schedule)}{CN * 0.8} \tag{12}$$

where

- $wl(schedule)$
- CN the number of all nodes confirming the schedule (including the creating node)

However, profit for mining the block $P(block)$ can be defined as follows:

$$P(block) = wl(block) * 0.2 \tag{13}$$

where

- $wl(block)$ the sum of $wl(schedule)$ for all transactions in the block

The proposed reward approach mainly depends on whether the blockchain is public or private. In the case of this approach, blockchain is public, but it can be freely changed to private. Therefore, the profits model may also vary depending on the given implementation, improved version of profits model was proposed in [50].

4.8 Evaluation Examples

In this section, we present the exemplary results of a simple experimental study, where 4 different cloud schedulers were implemented along with the blockchain algorithms[11]

We consider the following simple scenario:

1. The task scheduling process is run on 4 different scheduling modules; each module uses one of the RR, FCFS, HSGA or SJF algorithms (detailed description of the algorithms can be found in [29]).
2. BS is launched and 16 nodes are defined in the BC network. Each of them uses one of the four algorithms used by the scheduling modules described in point 1 to prepare the schedule. As a result, BS saves the best schedule selected by the network in the blockchain.
3. Schedules from points 1 and 2 are carried out for different datasets (different number of tasks and different number of virtual machines).
4. Each of the prepared schedules is evaluated according to different metrics. Points 1 and 2 are repeated to evaluate the schedule in terms of the values of makespan, flowtime, economic cost and resource utilization; the security level of the schedule is not taken into consideration (expected SL = 0). The value of each criterion is calculated on the basis of the results of schedule execution using the CloudSim simulator[12].
5. Points 1 and 2 are repeated for different expected SL, taking into account the makespan as a schedule evaluation criterion.

The following test datasets were prepared with the task and virtual machines characteristics (workloads and computing capacities) together with the security demand and trust level parameters. Tasks characteristics ($[wl_1, \ldots, wl_n]$, $[sd_1, \ldots, sd_n]$) and virtual machines characteristics ($[cc_1, \ldots, cc_m]$, $[tl_1, \ldots, tl_m]$) were generated according to the Gaussian distribution:

$$\mathcal{N}(\mu, \sigma^2) \tag{14}$$

where:

- μ mean
- σ^2 variance of random variable

In the literature, there are many cases where test data is generated for several different scenarios (different number of machines and tasks). The number of virtual machines usually ranges between 32 and 256, while the number of tasks from 512 to 4096. Kołodziej [28] provided four different scenarios where different numbers of tasks and machines were defined and data was generated according to the normal distribution for machines $\mathcal{N}(5000, 875)$ and for tasks

[11] A comprehensive experimental analysis with a lot of scenarios and full statistical analysis is presented in https://doktoraty.iet.agh.edu.pl/_media/2020:awilcz:phd_thesis_aw.pdf.

[12] http://www.cloudbus.org/cloudsim/.

Table 1. Characteristics of virtual machines

	Dataset 1	Dataset 2
Number of VM	32	128
Distribution of computing capacity (*cc*) values	$\mathcal{N}(7, 4)$	
Measure of *cc*	MFLOPS	
Minimum value of *cc*	1	
Maximum value of *cc*	12	
Distribution of trust level (*tl*)	$\mathcal{N}(0.5, 0.04)$	
Minimum value of *tl*	0.2	
Maximum value of *tl*	1	

Table 2. Characteristics of tasks

	Dataset 1	Dataset 2
Number of tasks	1024	4096
Distribution of workloads (*wl*) values	$\mathcal{N}(600, 90000)$	
Measure of *wl*	MFLO	
Minimum value of *wl*	100	
Maximum value of *wl*	1000	
Distribution of security demand (*sd*)	$\mathcal{N}(0.8, 0.0225)$	
Minimum value of *sd*	0.6	
Maximum value of *sd*	0.9	

$\mathcal{N}(250000000, 43750000)$, respectively. The real characteristics of the computational units can be found on the webpages [2–4] where different processors are compared. Taking account of the experiments conducted by Kołodziej, it was decided to generate 2 different datasets of characteristics of virtual machines and 2 different datasets of characteristics of tasks, whose key parameters are presented in Tables 1 and 2. Datasets were generated applying the Commons Math library [15] using the *NormalDistribution* class. Due to the fact that 2 datasets of characteristics of tasks and 2 datasets of characteristics of virtual machines were generated, each task scheduling process took place on 4 different datasets:

- 32 VM and 1024 tasks;
- 32 VM and 4096 tasks;
- 128 VM and 1024 tasks;
- 128 VM and 4096 tasks.

The main configuration parameters of the extended CloudSim presented in Tables 3 and 4.

Table 3. BCSchedCloudSim configuration

Number of records in the pool of requests	96
MTC	8
B_{wl}	1000000 MFLO
Time for which TF of validators is determined	30 days
Initial value of SF of each node in the BC network	583329 MFLO
Failure coefficient α (see Eq. 15)	2.5

Table 4. CloudSim configuration

1 data center	
Size of VM image	10000 MB
Memory of VM	4096 MB
Number of CPUs in VM	4

Each schedule execution was repeated 48 times. Then, the obtained results were compared with those returned by BS, where 96 requests with the same data were placed.

Evaluation of the Secure Blockchain Scheduler Based on Makespan [50]

According to the previously presented scenario, an experiment was carried out where makespan was adopted as a criterion for schedule evaluation. Size of the BC network was set to 16 nodes. In this part of experiments, the security level criterion was omitted and will be considered in the next section. The algorithm for calculating the value of makespan was presented in Eq. 1. The results of this experiment are shown in Fig. 17. The shorter the time necessary to execute the entire schedule, the better the schedule was. As demonstrated in the obtained results, in most cases the best result was achieved by BS. In three cases, the best makespan was returned. In one case, the returned makespan was not the best but it is not the worst either. Discrepancy between the minimum and maximum values returned by the scheduler is the smallest in the case of BS, which confirms the stability of its operation.

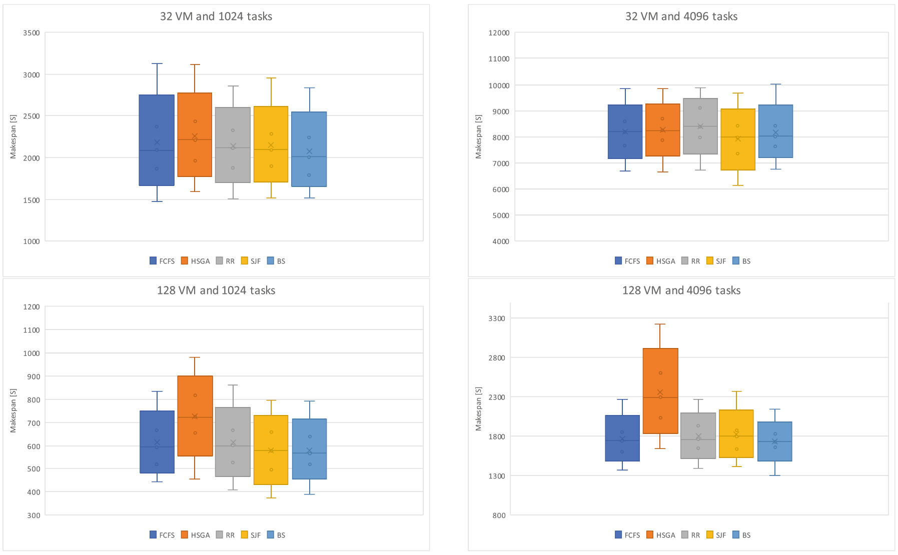

Fig. 17. Evaluation of the model performance using makespan Source: [49]

Evaluation of the Secure Blockchain Scheduler Based on the Security Level of the Schedule [51]

In the previous section, the security level criterion was omitted. In this section, the results of experiment assessing primarily this aspect are presented. Table 5 present the parameters of the considered scheduling scenarios.

Table 5. Experiments outline - expected security level equal to 1.5

SC	Expected SL	Size of the BC network
makespan	1.5	16 nodes

The SL was calculated based on the values of $P^{failure}$, P^{fake} and $P^{hacking}$ parameters [51].

$P^{failure}$ determines the probability of machine failure during task execution due to the high security requirements specified for this task. This factor is estimated the same for each scheduler used in the experiments. $P^{failure}_{i,j}$ for specific machine i and task j was defined by Kołodziej [28] as follows:

$$P^{failure}_{i,j} = \begin{cases} 0 & sd_j \leq tl_i \\ 1 - e^{-\alpha(sd_j - tl_i)} & sd_j > tl_i \end{cases} \tag{15}$$

where

- α failure coefficient defined as a global parameter
- sd_j described in Eq. 5
- tl_i described in Eq. 6

P^{fake} defines the probability that the schedulers send a false or incorrectly prepared schedule. The following schedulers – (FCFS, HSGA, RR and SJF) – were not intgrated with the blockchain algorithms, and in such cases we set the parameter P^{fake} to 0.5. In the case of blockchain-based scheduler BS, where the schedule is checked by other nodes from the network, the value of P^{fake} is calculated by using the following formula:

$$P^{fake} = 1 - \frac{N_c}{N_v} \tag{16}$$

where

- N_c number of all confirmations that the schedule is correct obtained by the node from the verification nodes
- N_v number of all verification of the schedule regardless of whether the answer was positive or negative

Assuming that there are 16 nodes in the network and MTC is equal to 8, it can be seen that in the case of BS this factor assumes a value within the range $[0, 0.5]$, depending on how many requests the node must send to the network to receive 8 confirmations.

The value of the last factor $P^{hacking}$ indicates the probability of manipulation or modification of the prepared schedule by unauthorized entities. In the case of a single schedule module (FCFS, HSGA, RR and SJF), it can be assumed that $P^{hacking}$ is equal to 0.5. For BS, this value depends on TF_t of the attacking node. The higher the TF_t the block adding node has, the greater the probability that it may launch a majority attack [43] to modify parts of the blockchain. The $P^{hacking}$ value in that case is calculated according to the following formula:

$$P_t^{hacking} = \begin{cases} 0.5 & TF_t \geq \frac{1}{2}BW_t \\ \frac{TF_t}{BW_t} & TF_t < \frac{1}{2}BW_t \end{cases} \tag{17}$$

where

- TF_t the sum of $wl(schedule)$ for all schedules added to the blockchain by attacking node within a given period of time t (in this simulation t is equal to 30 days)
- BW_t

The Eq. 17 shows that $P^{hacking}$ in the case of BS assumes a value within the range $[0, 0.5]$, depending on how high the attacker's trust factor is in the moment of adding the block to the blockchain.

Provided experiments show how much advantage BS module has over the other scheduling modules when it comes to the security level criterion. The results of these experiments are presented in Figs. 18 and 19. As can be seen, the difference in the number of virtual machines and tasks is not significant because the results are very similar. However, it is worth noticing that BS has a significant advantage over the other modules. In the case of BS these are values around 2 while in the case of other schedulers the values are close to 1.5. In the simulation, the expected SL was set to 1.5 and all schedulers managed to achieve such a result.

It could be also observed, that the security level of the schedule prepared by BS is about 0.5 larger than in other cases. The effect of changing the number of nodes and MTC in the BC configuration on the P^{fake} value was checked. The results demonstrate that the P^{fake} value raises with the number of nodes in the network, which means that for larger networks there is a higher transaction rejection rate before it receives the appropriate number of confirmations. The impact of the chain of blocks size, specifically the BW_t value, on the $P^{hacking}$ factor. It turns out that the more transactions and blocks are in the chain, the lower the $P^{hacking}$ value is, which means that the security level of schedules grows with the length of blockchain and number of transactions placed in it. The highest SL achieved value was 2.266, a very good result compared to the 1.518 obtained by the other modules.

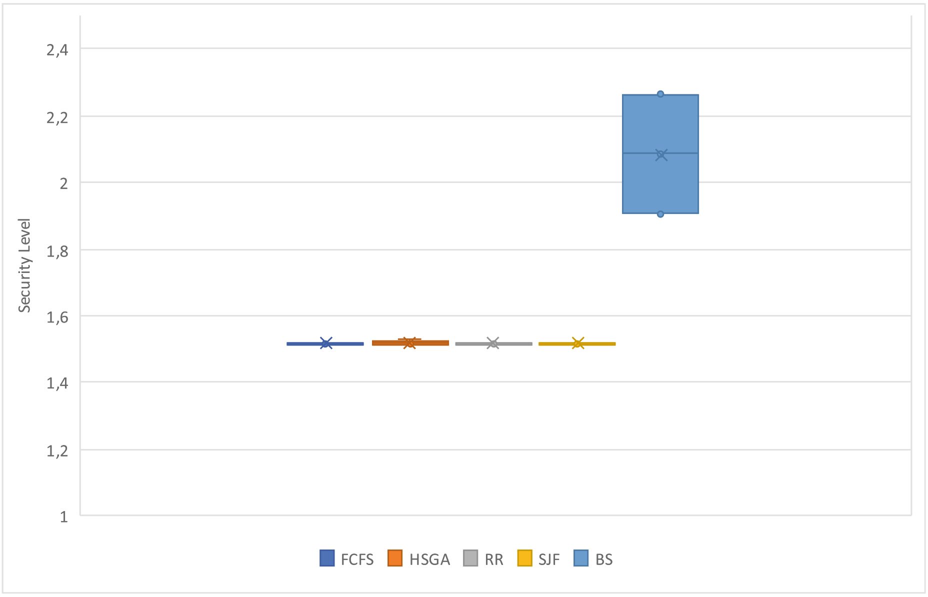

Fig. 18. Evaluation of the model performance using security level - 32 VM and 1024 tasks Source: [49]

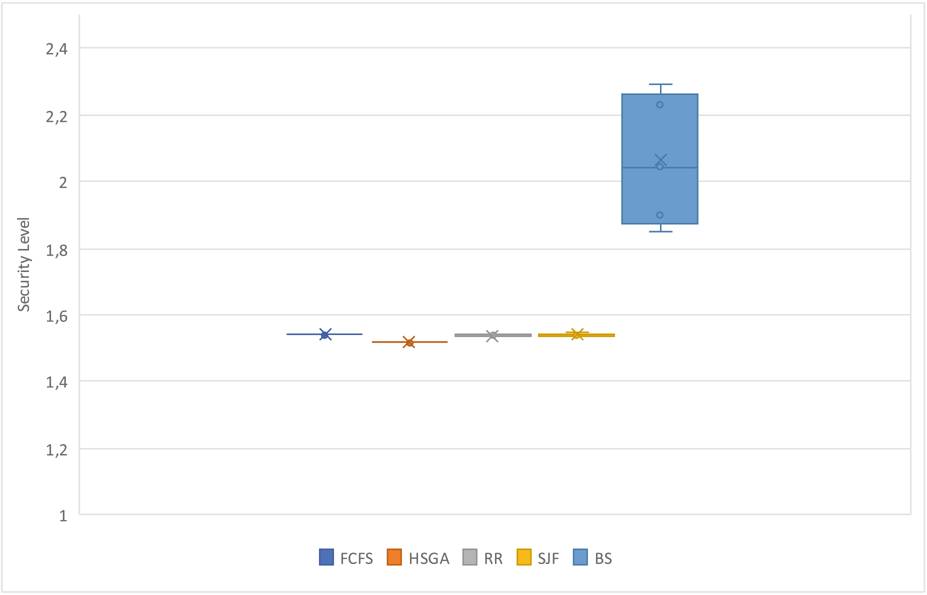

Fig. 19. Evaluation of the model performance using security level - 128 VM and 4096 tasks Source: [49]

5 Conclusions

Blockchain is a popular financial technology that uses ICT environments for virtual financial transactions using cryptocurrencies (e.g. Bitcoin). Blockchain customers store their transaction records in the blockchain P2P network, which effectively utilises the computing resources of its peers. A proof of work and a proof of stake are blockchain consensus algorithms used to improve the security of blockchain transactions.

This chapter briefly discussed the benefits of integrating the blockchain network with the elastic, scalable cloud environment to enhance the trustfulness of data servers and the security of data and user management. We also identified the challenges posed by this integration process. We presented a new concept of integrating the clouds infrastructure and schedulers with the blockchain algorithms to monitor the execution of the security-aware task scheduling in the cloud, one of the most important research topics in today's cloud and fog computing. We believe that the new blockchain-based scheduling model will allow us to overcome the problem of implementing the existing models in real-life scenarios.

Acknowledgements. Joanna Kolodziej's work was supported in part by the GUARD European Commission project under Grant Agreement no. 833456.

References

1. 51% attack (2019). https://www.investopedia.com/terms/1/51-attack.asp
2. CPU performance (2019). https://setiathome.berkeley.edu/cpu_list.php
3. Geekbench 5 - cross-platform benchmark (2019). https://www.geekbench.com/
4. Passmark - CPU benchmarks - list of benchmarked CPUs (2019). https://www.cpubenchmark.net/cpu_list.php
5. State of the dapps - ranking the best ethereum dapps (2019). https://www.stateofthedapps.com/rankings/platform/ethereum
6. What are blockchain confirmations? (2019). https://www.ethos.io/what-are-blockchain-confirmations
7. Discover our data center locations (2020). https://www.google.com/about/datacenters/locations/
8. Global infrastructure (2020). https://aws.amazon.com/about-aws/global-infrastructure/
9. Bošnjak, L., Sreš, J., Brumen, B.: Brute-force and dictionary attack on hashed real-world passwords. In: 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), pp. 1161–1166 (2018). https://doi.org/10.23919/MIPRO.2018.8400211
10. Buyya, R., Yeo, C.S., Venugopal, S., Broberg, J., Brandic, I.: Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility. Future Gener. Comput. Syst. **25**(6), 599–616 (2009). https://doi.org/10.1016/j.future.2008.12.001
11. Chana, I., Kaur, T.: Delivering IT as A utility- A systematic review. CoRR abs/1306.1639 (2013). http://arxiv.org/abs/1306.1639
12. Chowdhury, S.: Survey on various scheduling algorithms. Imp. J. Interdiscip. Res. (IJIR) **3**, 4 (2018)
13. Ding, C.H., Nutanong, S., Buyya, R.: P2P networks for content sharing. CoRR cs.DC/0402018 (2004). http://arxiv.org/abs/cs/0402018
14. Dorovskaya, D.: Review of peercoin cryptocurrency: history of creation and predictions (2018). https://peercoin.net/
15. The Apache Software Foundation: Commons math: The apache commons mathematics library (2016). https://commons.apache.org/proper/commons-math/
16. Gagniuc, M.G.: Virtual machines technologies. In: Proceedings of 9th International Conference on Development and Application Systems, Suceava, Romania, pp. 200–205 (2008)
17. Garner, B.: What is decred (DCR)? — a guide on decentralized blockchain governance (2019). https://coincentral.com/decred-lowdown-decentralized-blockchain-governance/
18. Grzonka, D., Szczygiel, M., Bernasiewicz, A., Wilczynski, A., Liszka, M.: Short analysis of implementation and resource utilization for the openstack cloud computing platform. In: Mladenov, V.M., Georgieva, P., Spasov, G., Petrova, G. (eds.) ECMS 2015 Proceedings of European Council for Modeling and Simulation, Albena (Varna), Bulgaria (2015). https://doi.org/10.7148/2015
19. Guharoy, R., et al.: A theoretical and detail approach on grid computing a review on grid computing applications. In: 2017 8th Annual Industrial Automation and Electromechanical Engineering Conference (IEMECON), pp. 142–146 (2017). https://doi.org/10.1109/IEMECON.2017.8079578
20. Karatza, H.D.: Performance evaluation and analysis of large scale distributed systems: issues, trends, problems and solutions (2016). http://chipset-cost.eu/wp-content/uploads/2016/05/EK-SummerSchool-Bucharest-Sep2016.pdf

21. Hileman, G., Rauchs, M.: Global Cryptocurrency Benchmarking Study. Cambridge Centre for Alternative Finance, Cambridge Judge Business School, University of Cambridge (2017). https://EconPapers.repec.org/RePEc:jbs:altfin:201704-gcbs
22. Hong, Z., Wang, Z., Cai, W., Leung, V.C.M.: Blockchain-empowered fair computational resource sharing system in the D2D network. Future Internet **9**, 85 (2017)
23. Jakóbik, A., Grzonka, D., Palmieri, F.: Non-deterministic security driven meta scheduler for distributed cloud organizations. Simul. Model. Pract. Theory **76**, 67–81 (2017). https://doi.org/10.1016/j.simpat.2016.10.011. http://www.sciencedirect.com/science/article/pii/S1569190X16302532. High-Performance Modelling and Simulation for Big Data Applications
24. Joshi, A.P., Han, M., Wang, Y.: A survey on security and privacy issues of blockchain technology. Math. Found. Comput. **1**, 121 (2018). https://doi.org/10.3934/mfc.2018007. http://aimsciences.org//article/id/d27803a2-7ce7-46e8-900d-30001fd4785a
25. Kalra, M., Singh, S.: A review of metaheuristic scheduling techniques in cloud computing. Egypt. Inform. J. **16**(3), 275–295 (2015). https://doi.org/10.1016/j.eij.2015.07.001. http://www.sciencedirect.com/science/article/pii/S1110866515000353
26. Kar, U.N., Sanyal, D.K.: An overview of device-to-device communication in cellular networks. ICT Express **4**(4), 203–208 (2018). https://doi.org/10.1016/j.icte.2017.08.002. http://www.sciencedirect.com/science/article/pii/S2405959517301467
27. Khatibi Bardsiri, A., Hashemi, S.M.: QoS metrics for cloud computing services evaluation. Int. J. Intell. Syst. Technol. Appl. **6**, 27–33 (2014). https://doi.org/10.5815/ijisa.2014.12.04
28. Kołodziej, J.: Evolutionary Hierarchical Multi-Criteria Metaheuristics for Scheduling in Large-Scale Grid Systems (2012)
29. Kołodziej, J., Wilczyński, A.: Blockchain secure cloud: a new generation integrated cloud and blockchain platforms - general concepts and challenges. Eur. Cybersecur. J. **4**(2), 28–35 (2018)
30. Kołodziej, J., Xhafa, F.: Integration of task abortion and security requirements in GA-based meta-heuristics for independent batch grid scheduling. Comput. Math. Appl. **63**(2), 350–364 (2012). https://doi.org/10.1016/j.camwa.2011.07.038. http://www.sciencedirect.com/science/article/pii/S089812211100592X. Advances in context, cognitive, and secure computing
31. Li, Z., et al.: A security and cost aware scheduling algorithm for heterogeneous tasks of scientific workflow in clouds. Future Gener. Comput. Syst. **65**, 140–152 (2016). https://doi.org/10.1016/j.future.2015.12.014. http://www.sciencedirect.com/science/article/pii/S0167739X15003982. Special Issue on Big Data in the Cloud
32. Lin, I.C., Liao, T.C.: A survey of blockchain security issues and challenges. Int. J. Netw. Secur. **19**, 653–659 (2017)
33. Lokhandwala, F.A.: A Heuristic Approach to Improve Task Scheduling in Cloud Computing using Blockchain technology. Master's thesis, Dublin, National College of Ireland. http://trap.ncirl.ie/3299/
34. Madni, H., Shafie, A.L., Abdullahi, M., Abdulhamid, S., Usman, M.: Performance comparison of heuristic algorithms for task scheduling in IAAS cloud computing environment. PLoS ONE **12**, e0176321 (2017). https://doi.org/10.1371/journal.pone.0176321
35. Michael, J.W., Rennock, A.C., Butcher, J.: Blockchain technology and regulatory investigations. Practical Law (2018)

36. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system. Cryptography Mailing list (2009). https://metzdowd.com
37. Perrey, R., Lycett, M.: Service-oriented architecture. In: Proceedings of 2003 Symposium on Applications and the Internet Workshops, pp. 116–119 (2003). https://doi.org/10.1109/SAINTW.2003.1210138
38. Pointcheval, D.: Asymmetric cryptography and practical security (2002)
39. Prasanth, A.: Cloud computing services: a survey. Int. J. Comput. Appl. **46**, 975–8887 (2012)
40. Rachmawati, D., Tarigan, J., Ginting, A.B.C.: A comparative study of message digest 5(MD5) and SHA256 algorithm. J. Phys.: Conf. Ser. **978**, 012116 (2018). https://doi.org/10.1088/1742-6596/978/1/012116
41. Rani, B.K., Rani, B.P., Babu, A.V.: Cloud computing and inter-clouds - types, topologies and research issues. Procedia Comput. Sci. **50**, 24–29 (2015). https://doi.org/10.1016/j.procs.2015.04.006. http://www.sciencedirect.com/science/article/pii/S1877050915005074. Big Data, Cloud and Computing Challenges
42. Reeb, J.E., Leavengood, S.A.: Using the simplex method to solve linear programming maximization problems (1998)
43. Saad, M., et al.: Exploring the attack surface of blockchain: a systematic overview. arXiv abs/1904.03487 (2019)
44. Tasca, P., Tessone, C.: A taxonomy of blockchain technologies: principles of identification and classification. Ledger **4** (2019). https://doi.org/10.5195/ledger.2019.140. https://ledgerjournal.org/ojs/index.php/ledger/article/view/140
45. Taufer, M., Rosenberg, A.: Scheduling DAG-based workflows on single cloud instances: high-performance and cost effectiveness with a static scheduler. Int. J. High Perform. Comput. Appl. **31**, 19–31 (2015). https://doi.org/10.1177/1094342015594518
46. Vujičić, D., Jagodić, D., Randić, S.: Blockchain technology, bitcoin, and ethereum: a brief overview. In: 2018 17th International Symposium INFOTEH-JAHORINA (INFOTEH), pp. 1–6 (2018). https://doi.org/10.1109/INFOTEH.2018.8345547
47. Wang, S., Ouyang, L., Yuan, Y., Ni, X., Han, X., Wang, F.Y.: Blockchain-enabled smart contracts: architecture, applications, and future trends. IEEE Trans. Syst. Man Cybern.: Syst. **49**, 2266–2277 (2019). https://doi.org/10.1109/TSMC.2019.2895123
48. Wilczyński, A., Kołodziej, J.: Virtualization model for processing of the sensitive mobile data. In: Kołodziej, J., Pop, F., Dobre, C. (eds.) Modeling and Simulation in HPC and Cloud Systems. SBD, vol. 36, pp. 121–133. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-73767-6_7
49. Wilczyński, A.: Blockchain-based task scheduling in computational clouds. Ph.D. thesis, AGH University of Science and Technology (2020). https://doktoraty.iet.agh.edu.pl/_media/2020:awilcz:phd_thesis_aw.pdf
50. Wilczyński, A., Kołodziej, J.: Modelling and simulation of security-aware task scheduling in cloud computing based on blockchain technology. Simul. Model. Pract. Theory 102038 (2019). https://doi.org/10.1016/j.simpat.2019.102038. http://www.sciencedirect.com/science/article/pii/S1569190X19301698
51. Wilczyński, A., Kołodziej, J., Grzonka, D.: Security aspects in blockchain-based scheduling in mobile multi-cloud computing. In: 2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid), pp. 696–703 (2021). https://doi.org/10.1109/CCGrid51090.2021.00084

52. Wilczyński, A., Widłak, A.: Blockchain networks - security aspects and consensus models. J. Telecommun. Inf. Technol. **2**, 46–52 (2019). https://doi.org/10.26636/jtit.2019.132019
53. Xoxa, N., Zotaj, M., Tafa, I., Fejzaj, J.: Simulation of first come first served (FCFS) and shortest job first (SJF) algorithms (2014)
54. Yaga, D., Mell, P., Roby, N., Scarfone, K.: Blockchain technology overview. NIST Interagency/Internal Report (NISTIR) - 8202 (2018). https://doi.org/10.6028/NIST.IR.8202
55. Zheng, Z., Xie, S., Dai, H., Chen, X., Wang, H.: An overview of blockchain technology: architecture, consensus, and future trends. In: 2017 IEEE International Congress on Big Data (BigData Congress), pp. 557–564 (2017). https://doi.org/10.1109/BigDataCongress.2017.85