Security aspects in blockchain-based scheduling in mobile multi-cloud computing

Andrzej Wilczyński
Department of Computer Science
Cracow University of Technology
Cracow, Poland
andrzej.wilczynski@pk.edu.pl

Joanna Kołodziej
Research and Academic
Computer Network (NASK)
Warsaw, Poland
Cracow University of Technology
Cracow, Poland
jokolodziej@pk.edu.pl

Daniel Grzonka
Department of Computer Science
Cracow University of Technology
Cracow, Poland
dgrzonka@pk.edu.pl

***Abstract*—The intensive development and growth in the popularity of mobile cloud computing services bring a critical need to introduce new solutions that increase the level of cloud and users security. One of the critical issues in highly distributed computational systems is a task scheduling process. This process may be exposed to many external and internal security threats, like task injection, machine failure or generation of incorrect schedule. These problems are especially important in mobile environments. It can be even more complicated if we take into consideration the personalization of the services offered. Recently, blockchain has been gaining rapidly in popularity, combining high efficiency with applications in distributed and highly personalized computational environments. In this paper, we developed and described a novel model for security-aware task scheduling in cloud computing based on blockchain technology. Unlike other blockchain-based solutions, the proposed model uses Proof of Stake, which does not have high requirements for computing power. A series of conducted experiments confirmed the high efficiency of the proposed model.**

***Index Terms*—blockchain, mobile cloud computing, security, task scheduling, multi-cloud, Stackelberg game.**

I. Introduction

Over the past decade, Cloud Computing (CC) has become the leading technology for providing services, such as computing power, storage, and databases. The idea of cloud computing is to transfer the entire weight of IT services to the service provider and enable permanent remote access to subscribed resources.

The use of distributed systems has become a trendy and, above all, a practical approach to solving complex computing problems in many science fields. A particular development can be seen in mobile computing clouds and multi-clouds. The increase in popularity of such environments brings with it many challenges, including effective use of resources, ensuring an appropriate level of security, personalization of services or supervision over the entire environment's operation. These problems often require new, sophisticated and intelligent solutions.

Task scheduling is one of the most crucial processes in computational distributed environments. It comes down to the generation of an allocation map of tasks to the computing units. Scheduling is an essential element in a distributed system. In essence, it allows for appropriate use of available resources and can be a crucial issue in the computing environment's performance, e.g., computational clouds [1]. However, scheduling is not only limited to minimizing time and cost but also concerns such aspects as security, deadlines, energy consumption or deadlines for completing tasks. Such an important issue, especially in modern systems based on the idea of virtualization and personalization of resources, undoubtedly requires innovative solutions, ensuring an appropriate level of security and allowing for the implementation of the Service Level Agreement (SLA) [2] brokered between the providers and consumers.

Security-aware task scheduling is a complex process. It requires a comprehensive approach that takes into account both external and internal factors and expectations. In the process of schedule generation, we may encounter many risks, in particular: task injection, problems in ensuring data integrity and privacy, unauthorized modification of task results, failure of computing units, generation of false schedules, performing tasks on units with a defined level of security.

These challenges are especially crucial in mixed computing environments using mobile devices. On the other hand, we can observe a dynamic growth in the development of solutions based on the blockchain technology. The use of lightweight solutions that do not require large computing resources can be successfully used in mobile clouds.

The main contributions of this paper are the following:

- comprehensive critical analysis of the security–aware schedulers with particular emphasis on models dedicated to mobile environments and based on the blockachain technologies,
- development of the new model of the blockchain–based mobile multi-cloud scheduler – BSSch with new security demand and trust level parameters,
- experimental analysis of the developed model.

The rest of the paper is organized as follows. In Sec. II we briefly survey the existing security-aware cloud scheduling models with a special attention on those where blockchain technology is applied. Sec. III presents the backgrounds of the blockchain networks. In Sec. IV, we define our blockchain–

based security–aware scheduler, which is then experimentally examined in Sec. V. The paper ends with final remarks and future research plans in Sec. VI.

II. Related work

The task scheduling process in distributed large-scale environments is NP-complete and requires unique methods to generate the [3] solution. There have been many taxonomies, and reviews were characterising various approaches to solving this problem [4]–[7]. From the point of view of this research, the most important are independent task scheduling and security-aware scheduling.

In [8] the author proposed an independent task scheduling model based on ETC-matrix. ETC-matrix predicts the execution time of each task on each machine. Such estimation is used in the process of schedule generation. In other papers (e.g., [9], [10]), the author introduces security parameters, like trust level (tl) parameters defined for the resources and security demand (sd) specified for the tasks. These parameters describe abstractly the features that make up the safety of tasks and machines.

A. Jakóbik et al. in [11] presented an innovative security-aware meta scheduler controlled by genetic heuristics. The model is explicitly designed to prevent a task injection attack inside clouds. Additionally, the authors described two models for assuring users security level expectations. The first allows scheduling tasks only on the machines offering the proper security level. The second model takes into account the cryptographic operations necessary in the execution process of each task. These models are based on Kołodziej, and Xhafa papers [9]. In the experimental part, the influence of non-deterministic time intervals on the quality of generated schedules was examined.

Multi-Agent System based Cloud Monitoring (MAS-CM) model that supports the performance and security of tasks processing (especially gathering, scheduling and execution) in computational clouds is proposed in [12]. The proposed system is based on three types of intelligent agents equipped with supervision and voting mechanisms and support by decisions of artificial neural networks. Thanks to the agent support, the makespan was significantly reduced. The use of ANN allowed for sufficient identification of false tasks. One of the tasks of the agent system was to control the consistency of the processed tasks by using the hash function. The MAS-CM model has been significantly expanded and formalised at work [13].

Li et al. in [14] proposed a security and cost-aware task scheduling (SCAS) algorithm for clouds. They consider heterogeneous tasks processing based on workflow in cloud computing platform where the infrastructure can enable dynamic resource scaling on demand. SCAS is based on the meta-heuristic optimisation technique, particle swarm optimisation (PSO). The chosen optimisation objective is devised to minimise the total workflow execution cost, taking into account the deadline and risk constraints. To protect the task's processing against snooping, alteration and spoofing attacks, the authors implemented three security modules: authentication, integrity and confidentiality. Each task is adapting all three security measures, with the security levels depending on the user's expectations. In the experiential part, the impact of security services and risk coefficient were examined. The results confirmed the effectiveness of the proposed algorithm (in comparison to other approaches).

There is also recent work on the use of blockchain technology in computational cloud processes. This approach is used by F. A. Lokhandwala in [15]. The paper is on improving task scheduling in cloud computing using Blockchain technology and heuristic approach. In the presented model, a decentralised blockchain network was used to optimise the resource allocation process in terms of reducing energy consumption and lowering costs on the part of the service provider. A load of data centres stored in blocks is checked using smart contracts [16]. Tasks are assigned to the least loaded data centres. Lokhandwala concluded that the use of the presented approach is not energy efficient in task assigning, but is more suitable for data storage. Finally, the author failed to implement a smart contract that would help choose the DC with the least load.

Another task scheduling model based on a smart contract was presented by J. Fan et al. in [17]. The authors contributed an idea of the smart contract of Ethereum used to design a kind of task scheduling strategy for the Vehicular Cloud Computing environment. The proposed method can guarantee the non-repudiation of task execution information. This research builds the private chain of Ethereum on simulation vehicles node. Next, it creates and deploys the smart contract. The authors shortly proof that the task scheduling strategy can be useful in a specific case. The idea is used for non-repudiation problem of task scheduling information under the AVC environment based on the Vehicle to Vehicle structure.

H. Baniata et al. in [18] have proposed PF-BTS model - an Ant Colony Optimization (ACO) algorithm in a Fog-enabled Blockchain-assisted scheduling model. The authors use Blockchain miners for ACO-based task scheduling and award miner nodes for their contribution in generating the best schedule. The model ensures that data, location, identity, and usage information about the environment are not exposed. The authors ensure that their solution produces the best results. Unfortunately, the results presented are questionable. The paper does not present reliable comparisons on the same sets of test data. At the same time, it incorrectly informs about the excessive demand for the computational capacity of the model that is the basis for this work.

There are a lot of papers on scheduling issues in mobile cloud environments. However, their main optimisation criteria are energy consumption or performance [19]–[22]. Unfortunately, there are no solutions focusing on such an important aspect as the security level in computational task processing. This underlines the novelty of the proposed model.

III. Security aspects in blockchain technologies

Blockchain (BC) became recently a promising technology in solving the complex problems in distributed computing, business and engineering systems and evolved from Bitcoin model developed by Nakamoto in [23]. BC can be defined as a distributed ledger of records, which contains transactions confirmed by the cryptographic digital signatures and are grouped into blocks. BC architecture is a fully distributed system with many local clusters, which can be centralized or also decentralized. In such decentralized model all important operations, processes and procedures (i.e., data processing, transactions) can be provided locally based on consensus algorithms used for the confirmations of each transaction, maintaining data consistency.

Each transaction in BC system must refer to some previous transactions. Therefore there is a possibility to trace and verify what has happened with the processed data.

The BC block contains the following data:

- bn –block id (number),
- $hash_c$ – hash of the current block,
- $hash_b$ – hash of the previous block,
- $timest$ – timestamp,
- l_t – list of transactions.

The detailed description of the block structure can be found in [24]. It can be noted that each block contains the hash of the previous block and the minor data modifications can change the hash parameters, which may entail the interception and rejection of those modifications by the other BC nodes. The data stored in BC system should be immutable and each entry in the ledger must be confirmed by the network. The connected blocks build the chains, which are the main structures of BC model.

BC transactions and data transmission are realized though security-aware BC procedures [25]. Data protection procedures in BC are executed with dynamically changed various parameters, which makes the malicious data injections or data damage difficult [26]. The security of transactions is ensured by the use of cryptographic methods. The end users can send the data to the BC system after receiving a private key from the system administrator. This private key is applied to generate a signature used then for the confirmation of the requested transaction and prevent any unauthorized changes.

The other security-related features of BC systems include the low-level access to the data, identification, management and control over the risks, system anomalies and security vulnerabilities. Also the local consensus procedures in distributed BC system make it difficult for external attacks [27].

BC-based schedulers can be effective in optimal resource allocation and computing task distribution in decentralized computing systems, as it was demonstrated in our previous work [28]. Such schedulers can support the validation of the whole system under the individual preferences and conditions defined by the end users and related to the protection of their data and personal files. The local character of transactions and the distributed BC architecture make BC schedulers also useful in edge and mobile computing.

IV. Blockchain–based security–aware scheduling model (BSSch)

The main concept of the Blockchain–based Security–aware Scheduling (BSSch) model presented in this paper was inspired by the generic model of the Secure Blockchain Cloud Scheduler (SBCS) defined in [28], [29] and Security Level Scheduling (SLS) model defined in [8]. We modified the SLS parameters and extended the overall scheduling flow process primarily defined in SBSC with additional verification of the security condition defined by the Eq. 3 below.

A. BSSch model – SBSC with Security Level parameters

The main concept of the BSSCh model is presented in Fig. 1. We modified the notation defined in [8]. Communication between clouds providers in BBSCh is based on the generic communication model in decentralized public blockchain networks, where no special protocols for information exchange are required. The main actors in our model are customers (end-users) and service providers. The customers formulate their security and computational resources requirements and send them to the cloud providers. End-users' requests are forwarded to the *Task Managers* (TMs) from multi clouds providers. Based on the knowledge about available resources, each TM selects those resources (machines) that can calculate and complete at least one task. Task is usually allocated at machines, which are characterized by the following parameters: (i) cc - computing capacity and (ii) tl - trust level. Each task is characterized by (i) wl - workload and (ii) tl - security demand parameters. TM also sets the value of the expected security level (ESL) parameter for the schedule. If ESL is equal to 0 then all proposed schedules prepared by the nodes will be accepted regardless of their security level SL, otherwise only those with SL greater than or equal to the value given by TM will be considered as valid.

After setting the characteristics of the tasks and machines and the ESL, all the parameters are defined as the request and then stored in the request pool. Nodes receive prepared requests and prepare the schedule according to their own scheduling algorithm. After its preparation, one of the optimization metrics, for instance makespan, is calculated. Then the obtained results along with the data from the request are placed in the transaction, which is then broadcasted over the blockchain network for confirmation. After obtaining the appropriate number of confirmations, the transaction is placed in the block. Before adding it, SL of the prepared schedule is calculated. If its value is equal to or greater than the ESL defined by TM, the adding process continues, otherwise transaction is skipped as further processing would be pointless. A transaction is considered as correct if it passes the validation with the Proof of Schedule (PoSch) algorithm described in [28], according to which the optimal schedule is selected. The PoSch algorithm takes into account one of the optimization criteria, for instance makespan or economic costs, calculated for the proposed schedules. As part of this work, the blockchain protocol regulating the rules for adding subsequent blocks to the chain has been extended to include

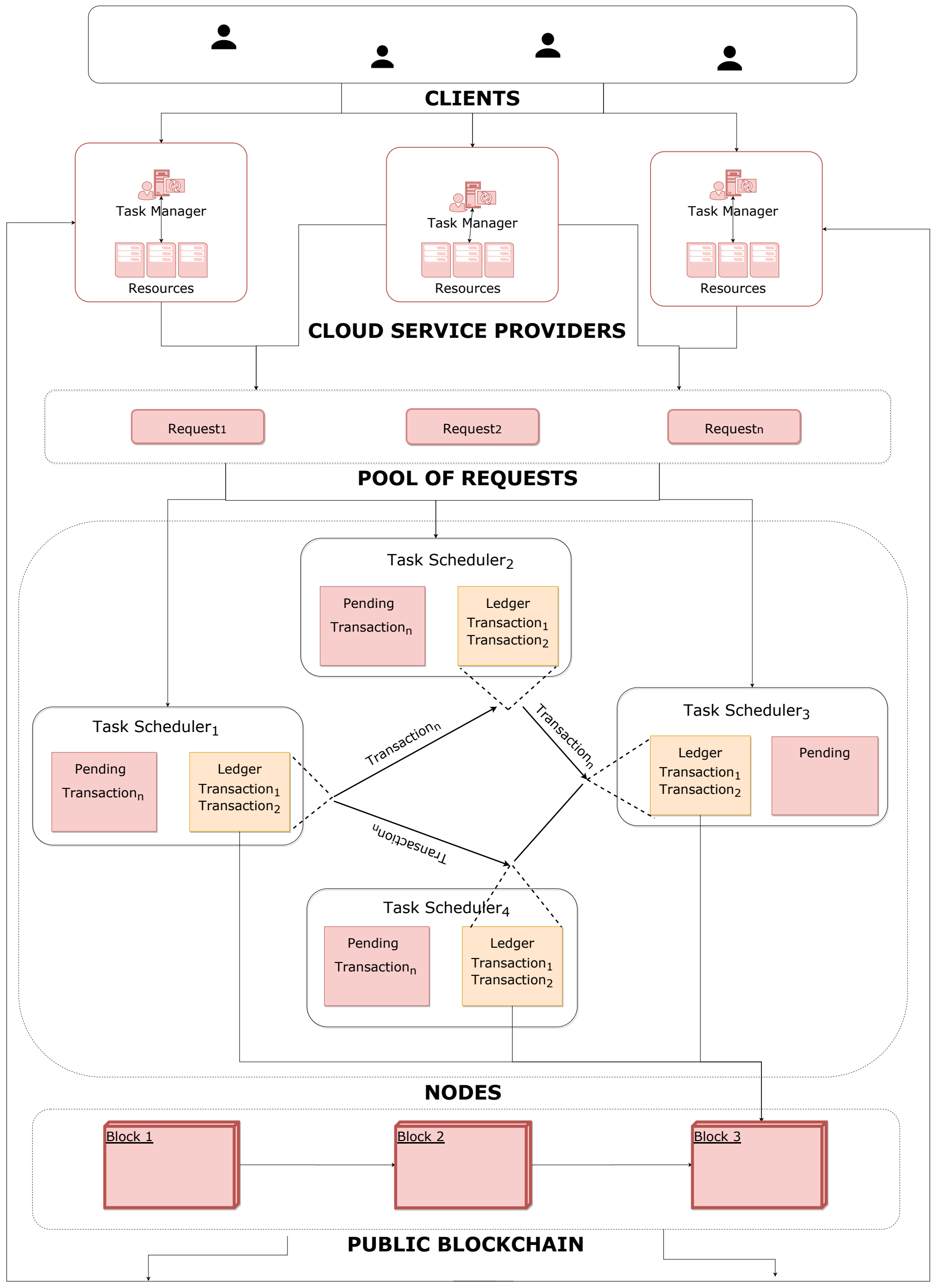

Fig. 1. BSSch model [28]

verification of the schedule also in terms of the security criterion defined as SL, which is calculated according to Eq. 8. After validation, the block with transactions is added to the blockchain by the validator node according to Proof of Stake (PoS) consensus algorithm [30]. When the new blockchain is confirmed throughout the BC network and recognized as official, TM's can download the schedule prepared for them. At this point, the whole process ends.

B. Modified SLS model

Our model is an extension of the model defined in [8]. In the current model, two main parameters, namely security demand (SD) for tasks and trust level (TL) for machines (mobile devices, virtual or physical machines and others) are defined in the following way:

$$SD_{n_{tasks}} = [sd_j, sd_{j+1}, \dots, sd_n] \tag{1}$$

where:

- $sd_j \in [0,1]$ is the security demand parameter defined for a given task j, which expresses security requirements for execution of that task in cloud resources, and
- n is the number of tasks in the batch.

$$TL_{m_{machines}} = [tl_i, tl_{i+1}, \dots, tl_m] \tag{2}$$

where:

- $tl_i \in [0, 1]$ – is the trust level parameter defined for i-th machine, which expresses the required trust level for that machine, and
- m is the total number of machines in the considered mobile cloud cluster (or all cloud).

In the above model, the following security condition:

$$sd_j \leq tl_i \tag{3}$$

means that the security requirements for a given task–machine pair $(j, i); i = 1, \ldots, n; i = 1, \ldots, n$ are satisfied and task j can be successfully executed on machine i. This model has been adjusted in terms of threats resulting from task processing in mobile cloud computing. Task level and security demand parameters are used for definition of the matrix $P^{failure}$ of probabilities of machines failures. $P^{failure}$ is the probability of machine failure during tasks execution due to high security restrictions. It could be, the inability to perform a given task on a given machine because it machine is located in an illegal geographic location or does not have a firewall. The probability of failure of machine i during the execution of the task j due to high security restrictions is calculated as follows:

$$P_{i,j}^{failure} = \begin{cases} 0 & sd_j \leq tl_i \\ 1 - e^{-\alpha(sd_j - tl_i)} & sd_j > tl_i \end{cases} \tag{4}$$

where α denotes the failure coefficient defined as a global parameter.

Let us denote by SL the security scheduling level. In order to define SL, we will need the following three parameters: $P^{failure}$, P^{fake} and $P^{hacking}$.

$P^{failure}$ for the complete schedule can be defined as follows:

$$P^{failure} = \frac{\sum_{i=1}^{m} \sum_{j \in Tasks_i} P_{i,j}^{failure}}{n} \tag{5}$$

where:

- i is the number of the machine,
- j is the number of the task,
- $Tasks_i$ is the set of tasks assigned to machine i,
- m is the number of machines,
- n is the number of tasks in the batch.

P^{fake} coefficient assumes that the node intentionally sending an incorrect schedule in order to only participate in building the block and get valuable assets or profits for it. P^{fake} is the probability that the false or incorrect schedule will be processed. The probability values are of the range $[0, 1]$, where 0 represents the lowest, and 1 represents the highest probability of schedule falsification. The value of P^{fake} is calculated as follows:

$$P^{fake} = 1 - \frac{N_c}{N_v} \tag{6}$$

where:

- N_c is a number of all confirmations that the schedule is correct received by the selected node from the other blockchain network nodes,
- N_v is a number of all verifications of the schedule, regardless of whether the the schedule was considered as valid or not by the other blockchain network nodes.

$P^{hacking}$ is the probability of manipulation, i. e. modification of the prepared schedule by unauthorized injections. The probability values are from the range $[0, 1]$, where 0 represents the lowest, and 1 represents the highest probability of task injections. To define $P^{hacking}$, the following notation is introduced:

- wl_j - workload - characterization of the task j,
- $wl(schedule)$ - the sum of vector elements $[wl_1, \ldots, wl_n]$, defined for all tasks from schedule

Based on the above notation, for BBSCh scheduler, $P^{hacking}$ parameter is calculated in the following way:

$$P_t^{hacking} = \begin{cases} 0.5 & TF_t \geq \frac{1}{2} BW_t \\ \frac{TF_t}{BW_t} & TF_t < \frac{1}{2} BW_t \end{cases} \tag{7}$$

where:

- TF_t is the sum of $wl(schedule)$ for all schedules added to the blockchain by attacking node within a given period of time t,
- BW_t is the sum of $wl(schedule)$ added to the blockchain within a given period of time t.

Having the above three parameters, SL can defined as follows:

$$SL = 3 - P^{failure} - P^{fake} - P^{hacking} \tag{8}$$

It can be observed that the values of SL are from the interval $[0, 3]$.

BSSch is used in mobile cloud computing to prepare schedule for launching resource-intensive mobile applications. Performing tasks in the correct order on available resources in such environment is very important to enable the execution of rich mobile applications on multiple mobile devices, with rich user experience. The model is also suitable for task scheduling in standard cloud computing environments.

V. Experiments

In this section we present the results of empirical evaluation of BBSCh model. The experiments were conducted in the BCSchedCloudSim simulation environment, which prototype was defined in [28]. This simulator implements four different task scheduling algorithms: First Come First Served (FCFS), Shortest Job First (SJF), Round Robin (RR) and Hybrid Heuristic Method based on Genetic Algorithm (HSGA). These algorithms were used by the nodes to prepare schedules (algorithms used by classic schedulers characterized by low computational complexity), while the CloudSim platform [31] was used to test their execution. The experiments presented a comparison of 4 modules using the above algorithms, where each of them operated independently without the use of the blokckchain technology and the fifth module called

	M–Dataset 1	M–Dataset 2
Number of machines	32	128
Distribution of cc values	$\mathcal{N}(7,4)$	
Measure of cc	MFLOPS	
Minimum value of cc	1	
Maximum value of cc	12	
Distribution of tl values	$\mathcal{N}(0.5,0.04)$	
Minimum value of tl	0.2	
Maximum value of tl	1	

TABLE I
CHARACTERISTICS OF MACHINES

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

TABLE II
CHARACTERISTICS OF TASKS

Blockchain Scheduler (BS) proposed in [28] and extended in this paper by security level of the schedule. This module uses blockchain technology to find a schedule according to the process described in Sec. IV. BS scheduler network is composed of 16 nodes, which may select one of the described scheduling algorithms for generating the schedules. Each of the 5 modules generates schedules for different input data (different number of tasks and different number of machines). This operation is repeated for different ESL, taking into account the makespan as an optimization criterion. Due to the low demand for computing capacity, these solutions are suitable for mobile cloud computing.

A. Simulation parameters

The methods of generating the parameters for test scenarios were similar than in our previous work with BCSchedCloudSim simulator and secure cloud scheduling experiments presented in [8]. We used workload (wl) ($[wl_1, \ldots, wl_n]$) and security demand (sd) ($[sd_1, \ldots, sd_n]$) parameters for task characteristics; and computing capacity (cc) ($[cc_1, \ldots, cc_m]$) and trust level (tl) ($[tl_1, \ldots, tl_m]$) parameters for machines characteristics. The values of all those parameters were generated by using the Gaussian distribution [8].

We considered in our experiment the following 4 scenarios for different number of tasks and machines:

- **Scenario I:** 32 machines and 1024 tasks;
- **Scenario II:** 32 machines and 4096 tasks;

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. 5)	2.5

TABLE III
BCSCHEDCLOUDSIM CONFIGURATION

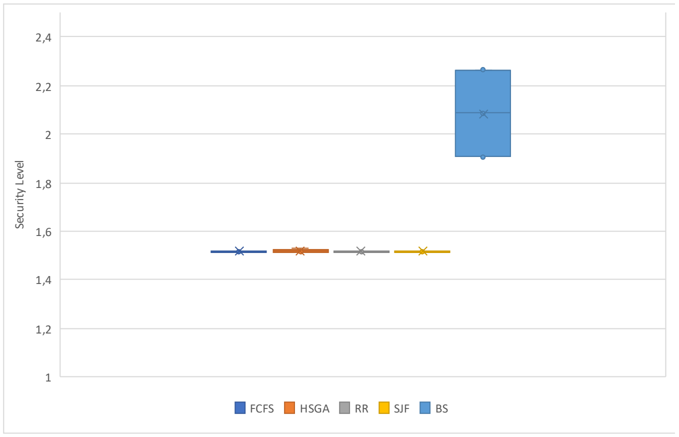

Fig. 2. Exp. 1 - security level for different schedulers - 32 machines and 1024 tasks

- **Scenario III:** 128 machines and 1024 tasks;
- **Scenario IV:** 128 machines and 4096 tasks.

Such scenarios reflect the situation of activation relatively small (32) and big (128) numbers of mobile resources (machines) used for small (1024) and big (4096) numbers of tasks arrived to the system. They should be considered through the prism of mobile processing of computing tasks.

For each considered number of machines and tasks we defined 4 separate datasets, namely M–Dataset 1, M–Dataset 2, T–Dataset 1, T–Dataset 2, by using the $NormalDistribution$ class in Commons Math library [32]. The values of the key parameters in those datasets are presented in Tables I and II. The main BCSchedCloudSim parameters are presented in Table III. For the BS module, each dataset was pushed 96 times as a request into the request pool. One of the available 16 nodes in the BC network, which initialized the transaction, must receive confirmation of the correctness of the schedule included in it, from 8 other nodes to consider the transaction as valid. Table V shows the parameters of the CloudSim simulator that were adopted during simulation. One datacenter with many resources is simulated each with 4096 MB of RAM and 4 CPUs, that is the characteristics of modern mobile devices. Each preparation of the schedule according to 4 independent modules was repeated 48 times. Then the averaged results were compared with those returned by BS, in which 96 requests with the same data were placed.

B. Experimental results

We divided our experiments into 7 groups. The main network parameters in each group of the experiments are presented in Tab. IV. The security level SL parameter used in the experiments was calculated according to Eq. 8. In that formula, $P^{failure}$ parameter is calculated based on the prepared schedule according to Eq. 5. The value of P^{fake} in formula 8 is 0.5 in the case of the implementation of independent schedule module (FCFS, HSGA, RR and SJF). In the case of BS (where schedule is verified by the other nodes), P^{fake} is calculated according to Eq. 6. The value of

						Dependent variables			
No.	ESL	Size of the BC network	MTC	Initial value of BW_t	Scheduler	$P^{failure}$	P^{fake}	$P^{hacking}$	SL
Exp. 1	1.5	16	8	2333316	FCFS	0.483	0.5	0.5	1.514
					HSGA	0.482	0.5	0.5	1.515
					RR	0.485	0.5	0.5	1.517
					SJF	0.484	0.5	0.5	1.519
					BS	0.484	0.1	0.325	2.091
Exp. 2	1.5	16	8	2333316	FCFS	0.484	0.5	0.5	1.517
					HSGA	0.482	0.5	0.5	1.507
					RR	0.481	0.5	0.5	1.518
					SJF	0.484	0.5	0.5	1.519
					BS	0.489	0.1	0.325	2,091
Exp. 3	2	4	2	2333316	BS	0,481	0	0,45	2,066
Exp. 4	2	8	4	2333316	BS	0,484	0,05	0,4	2,064
Exp. 5	2	12	6	2333316	BS	0,493	0,071	0,324	2,113
Exp. 6	2	16	8	4666632	BS	0,485	0,156	0,172	2,188
Exp. 7	2	16	8	6999948	BS	0,478	0,136	0,118	2,269

TABLE IV
EXPERIMENTS OUTLINE - VARIOUS SL AND BC NETWORK CONFIGURATION

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

TABLE V
CLOUDSIM CONFIGURATION

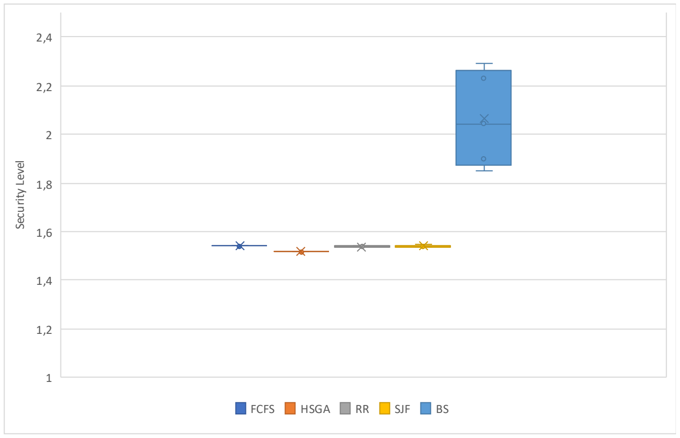

Fig. 3. Exp. 2 - security level for different schedulers - 128 machines and 4096 tasks

$P^{hacking}$ in the case of a independent schedule module (FCFS, HSGA, RR and SJF) is 0.5. For BS scheduler, this parameter is calculated according to Eq. 7.

The results of the Exp. 1 and 2 are presented in Fig. 2 and 3. As can be seen, the difference in the number of machines and tasks is not significant as the results are very similar. It is worth noting, however, that BS scheduler has significant advantages over other modules. In BS these are values around 2 while in the case of other schedulers the values are close to 1.5.

In subsequent experiments, simulations were carried out with various blockchain network configurations, omitting independent modules, to check the impact of blockchain network configuration on the SL of schedules prepared by BS. Makespan was used as the optimization criterion in each experiment. The number of machines was set to 32 and the number of tasks to 1024. In the experiments 3, 4 and 5, the impact of changing the number of nodes and the minimum number of transaction confirmations (MTC) in the blockchain configuration on the value of P^{fake} was checked. The results show that the value of P^{fake} increases with the number of nodes in the network, meaning that larger networks have a higher transaction rejection rate before it receives the appropriate number of confirmations. The last two experiments, 6 and 7, demonstrate the effect of blockchain size, namely the value of BW_t, on the $P^{hacking}$ factor. It turns out that the more transactions and blocks there are in the blockchain, the lower the value of $P^{hacking}$ is, which means that SL of schedules increases with the length of the blockchain and the number of transactions placed in it. The highest value of SL achieved was 2.269, which is a very good result compared to the 1.519 achieved by the independent modules.

VI. CONCLUSIONS

The main problems with most of the existing security-aware cloud schedulers are their high complexities. Execution of the security mechanisms usually is time consuming, which is also the main reason of high energy utilization in the cloud clusters. Such schedulers cannot be successfully implemented in mobile devices, where the limited battery lifetime is the main bottleneck of completion the security procedures and assigned tasks. The blockchain-based scheduling model presented in this paper is a promising alternate methodology to the other cloud schedulers with embedded security mechanisms. Fully distributed blockchain system, through cryptographic protocols in the consensus algorithms and local confirmation of the transactions, is recommended as a basic technology for development of the secure lightweight schedulers in mobile clouds. In Blockchain–based Security–aware Scheduling model (BSSch), we use simple Proof of Schedule algorithm, which significantly reduced the scheduler's complexity and made it even more suitable for the edge local and mobile computing systems, which is confirmed in the presented experimental analysis. Unlike the most blockchain-based solutions (based on Proof of Work consensus), the proposed model uses Proof of Stake consensus, which does not have high requirements for computing power.

The research presented in this paper is our first step towards the development of the open access blockchain-based cloud scheduling simulation environment, loosely inspired by CloudSim simulator and our previous implementations of the prototypes of blockchain schedulers. We plan to improve the core simulator by using the existing blockchain libraries and extend the list of the scheduling algorithms, as well as improve the security mechanisms. We also plan to define the realistic testbeds for testing the schedulers on the real data, which was problematic for most of the cloud schedulers with probabilistic-based security models.

Acknowledgment

The work of Joanna Kolodziej's was supported in part by the European Commission, under Grant Agreement no. 833456 (GUARD). The contribution of Daniel Grzonka was supported by the Foundation for Polish Science (FNP), under scholarship no. START 25.2019. The contribution of Andrzej Wilczyński was supported by the Foundation for Polish Science (FNP), under scholarship no. START 2021.

References

[1] Ashraf Zia and Muhammad Naeem Ahmad Khan. Identifying key challenges in performance issues in cloud computing. *International Journal of Modern Education and Computer Science*, 4(10):59, 2012.

[2] Damian Serrano, Sara Bouchenak, Yousri Kouki, Frederic Alvares de Oliveira, Jr., Thomas Ledoux, Jonathan Lejeune, Julien Sopena, Luciana Arantes, and Pierre Sens. SLA guarantees for cloud services. *Future Generation Computer Systems*, 54:233–246, 2016.

[3] M. R. Garey, D. S. Johnson, and Ravi Sethi. The Complexity of Flowshop and Jobshop Scheduling. *Mathematics of Operations Research*, 1(2):117–129, 1976.

[4] Ruby Annette. J, Aisha Banu. W, and Shriram Shriram. A Taxonomy and Survey of Scheduling Algorithms in Cloud: Based on task dependency. *International Journal of Computer Applications*, 82(15):20–26, November 2013.

[5] T. L. Casavant and J. G. Kuhl. A taxonomy of scheduling in general-purpose distributed computing systems. *IEEE Transactions on Software Engineering*, 14(2):141–154, 1988.

[6] Mala Kalra and Sarbjeet Singh. A review of metaheuristic scheduling techniques in cloud computing. *Egyptian Informatics Journal*, 16(3):275–295, 2015.

[7] Maria Alejandra Rodriguez and Rajkumar Buyya. A taxonomy and survey on scheduling algorithms for scientific workflows in IaaS cloud computing environments. *Concurrency and Computation: Practice and Experience*, 29(8):e4041, 2017.

[8] Joanna Kolodziej. *Evolutionary Hierarchical Multi-Criteria Metaheuristics for Scheduling in Large-Scale Grid Systems*, volume 419 of *Studies in Computational Intelligence*. Springer, 2012.

[9] Joanna Kołodziej and Fatos Xhafa. Integration of task abortion and security requirements in GA-based meta-heuristics for independent batch grid scheduling. *Computers & Mathematics with Applications*, 63(2):350 – 364, 2012.

[10] Daniel Grzonka, Joanna Kołodziej, Jie Tao, and Samee Ullah Khan. Artificial Neural Network support to monitoring of the evolutionary driven security aware scheduling in computational distributed environments. *Future Generation Computer Systems*, 51:72 – 86, 2015.

[11] Agnieszka Jakóbik, Daniel Grzonka, and Francesco Palmieri. Non-deterministic security driven meta scheduler for distributed cloud organizations. *Simulation Modelling Practice and Theory*, 76:67 – 81, 2017.

[12] Daniel Grzonka, Agnieszka Jakóbik, Joanna Kołodziej, and Sabri Pllana. Using a multi-agent system and artificial intelligence for monitoring and improving the cloud performance and security. *Future Generation Computer Systems*, 86:1106 – 1117, 2018.

[13] Daniel Grzonka. *Inteligentne systemy monitoringu procesów harmonogramowania w rozproszonych środowiskach dużej skali w ujeciu wieloagentowym*. PhD thesis, Polska Akademia Nauk. Instytut Podstawowych Problemów Techniki, Kraków, 2018. in Polish.

[14] Zhongjin Li, Jidong Ge, Hongji Yang, Liguo Huang, Haiyang Hu, Hao Hu, and Bin Luo. A security and cost aware scheduling algorithm for heterogeneous tasks of scientific workflow in clouds. *Future Generation Computer Systems*, 65:140 – 152, 2016.

[15] Fatema Akbar Lokhandwala. A Heuristic Approach to Improve Task Scheduling in Cloud Computing using Blockchain technology. Master's thesis, Dublin, National College of Ireland, 2018.

[16] Shuai Wang, Liwei Ouyang, Yong Yuan, Xiaochun Ni, Xuan Han, and Fei-Yue Wang. Blockchain-Enabled Smart Contracts: Architecture, Applications, and Future Trends. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 49:2266–2277, 02 2019.

[17] J. Fan, R. Li, and S. Li. Research on Task Scheduling Strategy: Based on Smart Contract in Vehicular Cloud Computing Environment. In *2018 1st IEEE International Conference on Hot Information-Centric Networking (HotICN)*, pages 248–249, 2018.

[18] Hamza Baniata, Ahmad Anaqreh, and Attila Kertesz. PF-BTS: A Privacy-Aware Fog-enhanced Blockchain-assisted task scheduling. *Information Processing Management*, 58(1):102393, 2021.

[19] S. Guo, J. Liu, Y. Yang, B. Xiao, and Z. Li. Energy-Efficient Dynamic Computation Offloading and Cooperative Task Scheduling in Mobile Cloud Computing. *IEEE Transactions on Mobile Computing*, 18(2):319–333, 2019.

[20] X. Lin, Y. Wang, Q. Xie, and M. Pedram. Task Scheduling with Dynamic Voltage and Frequency Scaling for Energy Minimization in the Mobile Cloud Computing Environment. *IEEE Transactions on Services Computing*, 8(2):175–186, 2015.

[21] Y. Li, M. Chen, W. Dai, and M. Qiu. Energy Optimization With Dynamic Task Scheduling Mobile Cloud Computing. *IEEE Systems Journal*, 11(1):96–105, 2017.

[22] Hua Peng, Wu-Shao Wen, Ming-Lang Tseng, and Ling-Ling Li. Joint optimization method for task scheduling time and energy consumption in mobile cloud computing environment. *Applied Soft Computing*, 80:534–545, 2019.

[23] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. *Cryptography Mailing list at https://metzdowd.com*, 03 2009.

[24] Andrzej Wilczyński and Adrian Widłak. Blockchain Networks – Security Aspects and Consensus Models. *Journal of Telecommunications and Information Technology*, 2:46–52, 07 2019.

[25] Archana Prashanth Joshi, Meng Han, and Yan Wang. A survey on security and privacy issues of blockchain technology. *Mathematical Foundations of Computing*, 1:121, 2018.

[26] 51% Attack. https://www.investopedia.com/terms/1/51-attack.asp, 10 2019.

[27] Joanna Kołodziej and Andrzej Wilczyński. Blockchain Secure Cloud: a New Generation Integrated Cloud and Blockchain Platforms – General Concepts and Challenges. *European Cybersecurity Journal*, 4(2), 07 2018.

[28] Andrzej Wilczyński and Joanna Kołodziej. Modelling and simulation of security-aware task scheduling in cloud computing based on Blockchain technology. *Simulation Modelling Practice and Theory*, 99:102038, 2020.

[29] Andrzej Wilczyński. *Blockchain-based task scheduling in computational clouds*. PhD thesis, AGH University of Science and Technology, 2020.

[30] Wai Thin, Naipeng Dong, Guangdong Bai, and Jin Dong. Formal Analysis of a Proof-of-Stake Blockchain. pages 197–200, 12 2018.

[31] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, César A. F. De Rose, and Rajkumar Buyya. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw., Pract. Exper.*, 41:23–50, 2011.

[32] The Apache Software Foundation. Commons Math: The Apache Commons Mathematics Library. https://commons.apache.org/proper/commons-math/, 2016.