Computers and Electrical Engineering 92 (2021) 107087

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

Scalable anomaly detection in blockchain using graphics processing unit☆

Shin Morishima

Faculty of Engineering, Toyama Prefectural University, 5180 Kurokawa, Imizu-shi, Toyama, 939-0398, Japan

ARTICLE INFO

Keywords:
Blockchain
Anomaly detection
Graphics processing unit
Parallel computing
Bitcoin

ABSTRACT

In blockchain, approved transactions, including illegal ones, cannot be modified unlike existing bank transactions. To prevent the damage caused by illegal transactions, rapid anomaly detection of transactions is required because transactions can be modified before approval. However, existing anomaly detection methods must process all transactions in blockchain, and the processing time is longer than the interval of each approval. In this paper, we propose a subgraph-based anomaly detection method to perform the detection using a part of the blockchain data. The proposed structure of the subgraph is suitable for graphics processing units (GPUs) to accelerate detection by using parallel processing. In an evaluation using real Bitcoin transaction data, when the number of targeted transactions was one hundred, the proposed method was 11.1x faster than an existing GPU-based method without lowering the detection accuracy.

1. Introduction

Blockchain is an electronic cash system proposed by Bitcoin [1] that enables money transfer without a trusted third party. The main applications of blockchain are interpersonal and international transactions, which take advantage of the low cost of excluding fees for a trusted third party, such as a bank. Moreover, blockchain has features such as fault tolerance, tamper resistance, and anonymity. Blockchain is expected to be used for non-currency asset transactions [2,3], distributed applications [4,5], document storage systems, and land registrations due to these features. Blockchain consists of a chain structure connected by hash values of each block. No one (even in the transaction creator) can update or delete a blockchain transaction with this structure. This means the structure has high tampering resistance. However, it also means that correction of fraudulent transactions created by stolen private keys is not possible. Therefore, appropriate countermeasures are required to suppress the damage, such as the rapid detection of illegal transactions and the correction of such transactions before approval. A graph-based anomaly detection method for blockchain transactions using K-means clustering, local outlier factor [6], and Mahalanobis distance was proposed by Pham et al. [7]. However, this method must process all blockchain transactions because it can only detect an anomaly from the entire blockchain. The number of blockchain transactions increases as the system operates, and the execution time of this method is proportional to the number of transactions. Moreover, anomalous transactions, which can be detected by certain features and algorithms, are limited because each one has different features. Thus, anomaly detection should be repeated while the feature quantities and algorithms are varied. This repetition causes heavy computations and long execution time. An acceleration of blockchain anomaly detection using the graphics processing unit (GPU), specialized for parallel computing, was proposed in our earlier work [8]. In this work, the execution time of anomaly detection was shortened. However, the problem of increasing computation burden and cost in proportion to the number of blockchain transactions was not addressed.

☆ This paper is for CAEE special section VSI-pdcat3. Reviews processed and recommended for publication to the Editor-in-Chief by Guest Editor Hui Tian.

E-mail address: morisima@pu-toyama.ac.jp.

https://doi.org/10.1016/j.compeleceng.2021.107087
Received 30 March 2020; Received in revised form 5 November 2020; Accepted 9 March 2021
Available online 25 March 2021

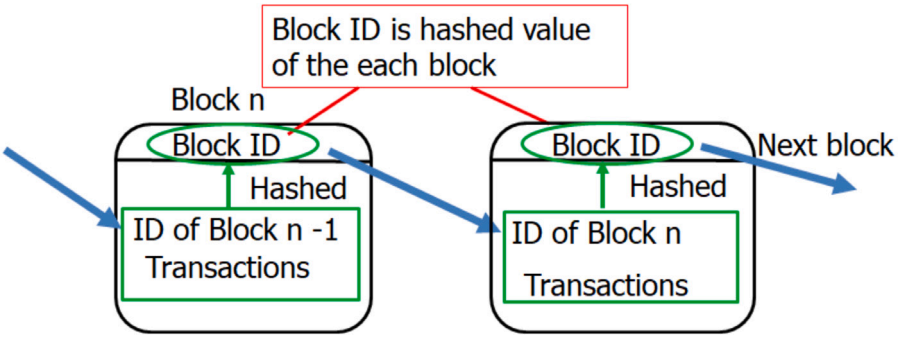

Fig. 1. Blockchain structure.

Therefore, in the present study, we propose a subgraph-based anomaly detection method to prevent the increase in processing time. The subgraph is localized by a targeted transaction for anomaly detection. By using the subgraph, the computation cost of anomaly detection becomes constant regardless of the number of blockchain transactions. Specifically, we propose method for making an overall graph from original blockchain transactions and a method for making a subgraph from the overall graph. Moreover, we propose a subgraph structure suitable for GPU processing to accelerate anomaly detections with parallel processing. In this structure, the creation of the subgraph, feature extraction, and anomaly detection are performed on the GPU. However, the target number of transactions in the subgraph-based method is limited to one. In blockchain systems, all transactions in a block are added simultaneously, and anomaly detection may be required for all of these transactions. Thus, we also propose an extended subgraph-based method for multiple transactions, which merges subgraphs. Based on these proposals, multiple anomaly detection for blockchain can be performed during approval intervals.

The rest of this paper is organized as follows. Section 2 details related work. Section 3 describes our proposed subgraph-based anomaly detection method using GPU for blockchain transactions and Section 4 presents an extension of the proposed method for multiple transactions. Section 5 presents the evaluation results and Section 6 discusses the application of the method. Section 7 concludes the paper.

2. Related work

2.1. Blockchain structure

An overview of the blockchain structure is shown in Fig. 1. A block consists of the block ID, ID of the previous block, and transactions. A transaction includes the senders' (inputs) and recipients' (outputs) addresses and the amount of remittance. A block ID is calculated by hashing the contents of the block, and the subsequent block includes the block ID. The relationship between a block with the block ID continues to the first block of the blockchain. Changes in the transaction content lead to changes in all the block IDs owing to their dependent relationship. When an attacker tampers with a transaction, all the block IDs must be tampered, which indicates blockchain's tamper resistance. The addition of new data for blockchain is performed by creating a new block, and the new block is attached as the latest block in the chain. A network consensus based on one of the several available consensus algorithms (e.g., Proof of Work (PoW) and Proof of Stake (PoS)) [9] is required to create a new block.

No one, even the transaction creator, can correct a blockchain transaction after the block is added to the blockchain by the consensus algorithm. Furthermore, fraudulent transactions derived from the theft of secret keys are not subject to a protocol and cannot be canceled at the time of addition of the block by the consensus algorithm. However, there is a time period before transaction approval by the consensus algorithm after its creation. Fraudulent transactions can be corrected if they are detected before the approval. In this paper, we propose a method of anomaly detection with high speed and accuracy to limit the damage caused by illegal transactions.

2.2. Anomaly detection on blockchain

Generally, blockchain anomaly detection is classified into two ways; (1) overall network anomaly detection (e.g., a blockchain network division or a fork of blockchain [10]) and (2) detection of fraudulent transactions created by operation missing or stolen secret keys, denoted as anomaly detection. The second one is the subject of this paper. Hereafter, anomaly detection refers to the latter.

A method for anomaly detection of blockchain transactions was proposed by Pham and Lee [7], who used a user-centric graph indicating the remittance flow between users. Currently, this structure (Fig. 2) is one of the typical structures in blockchain analysis [11]. In this paper, the user-centric graph made by overall blockchain transactions is called "UserGraph" to distinguish it from our proposed subgraph-based graph.

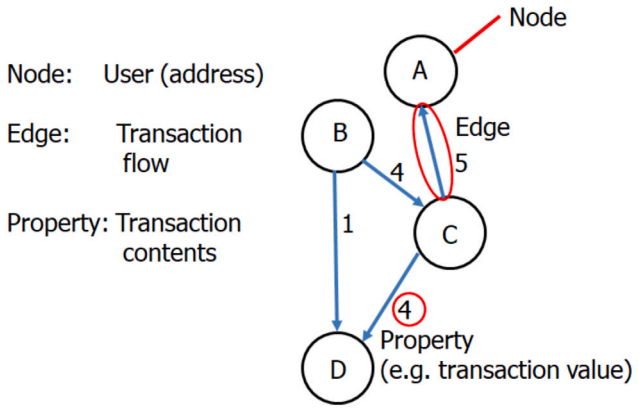

Fig. 2. Overview of UserGraph.

Based on Fig. 2, the developed UserGraph consists of three elements; nodes, edges, and properties. In a UserGraph, a node represents the user, an edge represents the remittance flow between users, and a property indicates the content of a transaction. Because an original transaction can have multiple inputs and outputs, an original transaction may include multiple remittance flows. Thus, a transaction in UserGraph is distinguished from an original transaction and represents remittance flow. The feature quantities of each user are extracted from the UserGraph. For instance, the number of outgoing edges shows the number of withdrawal transactions, and the number of ingoing edges in a certain node shows the number of deposit transactions. Three anomaly detection algorithms were used; K-means clustering, local outlier factor [6], and Mahalanobis distance. Two real theft cases from 2011 on the Bitcoin network were successfully detected by this method. The detected theft cases changed depending on the anomaly detection algorithm and the feature quantities, implying that the anomaly detection should be repeated by changing the feature quantities or the algorithm. The UserGraph-based methods [7] can perform anomaly detection for multiple transactions at the same time. Therefore, it is useful for historical transactions analysis without time constraints as in the example above.

The number of all the blockchain transactions is large, and the computation cost for anomaly detection algorithms increases with the number of transactions. In our earlier work, we proposed a GPU-based acceleration method for the detection of anomalies; this method was 32.6x faster than the CPU-based method [8]. However, the amount of calculations increases with the number of transactions, indicating that the previous method lacks scalability.

3. Subgraph-based anomaly detection method using GPU

3.1. Overview of proposed method

Fig. 3 illustrates the overall structure of the proposed method divided into the CPU side and GPU side. At first, the UserGraph is created from original transactions on the CPU side. The UserGraph is then sent to the GPU and is cached in the GPU. When the anomaly detection is requested for the GPU side, the subgraph is created in the GPU. If multiple transactions are targeted, as shown in the figure, the subgraphs are merged during subgraph making. The anomaly detection, which includes a feature extraction and anomaly detection algorithm, is performed simultaneously for multiple transactions using the merged subgraph. If a single transaction is targeted, the subgraph is not merged and directly used for anomaly detection. The merged subgraph and the targeted multiple transactions form is the extension based on the case of targeting a single transaction. This section describes the case of anomaly detection for a single transaction, and the extension is described in Section 4. The following subsections describe each process in detail.

3.2. UserGraph making considering blockchain features

Typically, a unique UserGraph with each node corresponding to one address cannot be created by using an original blockchain transaction. Because the transaction has multiple input and output addresses (Fig. 4(left)), multiple graphs result from their combination. Thus, in this study, all input addresses of the same transaction are combined and regarded to originate the same user before creating the graph. In a blockchain system, each user can freely generate multiple addresses. If the user has the secret key, he is identified as the owner of an address. A transaction creation must be accompanied by the secret key of the input address. If there are multiple input addresses in the same transaction, the transaction creation requires the keys of all the input addresses. Thus, the input addresses in the same transaction can be regarded to belong to the same user.

Regarding other blockchain features, raw transaction information includes self-transaction and one-time addresses, which are disposable for the purpose of anonymity [11]. These may reduce the accuracy of anomaly detection, and their deletion does not change the meaning of the transaction. Hence, after creation of an UserGraph, self-transactions are deleted (Fig. 5, Step 1). One-time addresses are detected by the number of inputs and outputs and the balance. If an address is a one-time address, the number of inputs and outputs is 1, and the balance of the address is 0. If an address meets the two conditions, it is deleted (Fig. 5, Step 2).

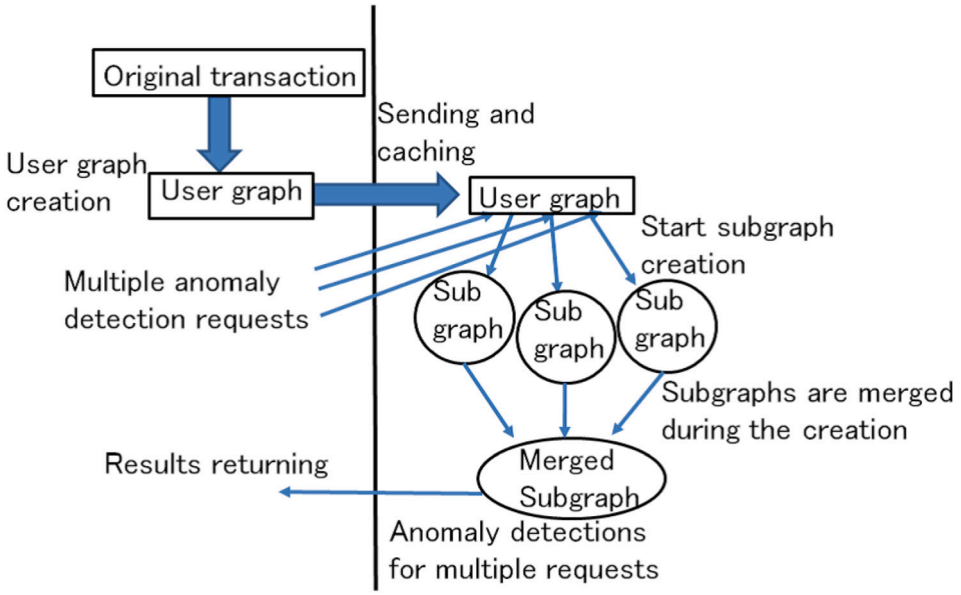

Fig. 3. Overall structure of proposed method.

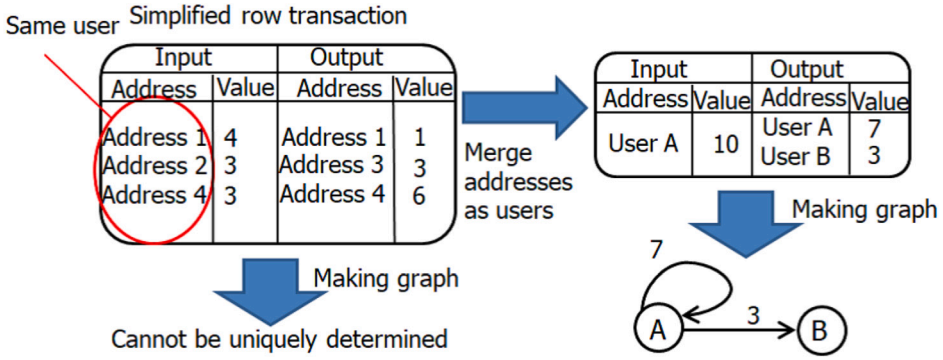

Fig. 4. Example of merging input addresses.

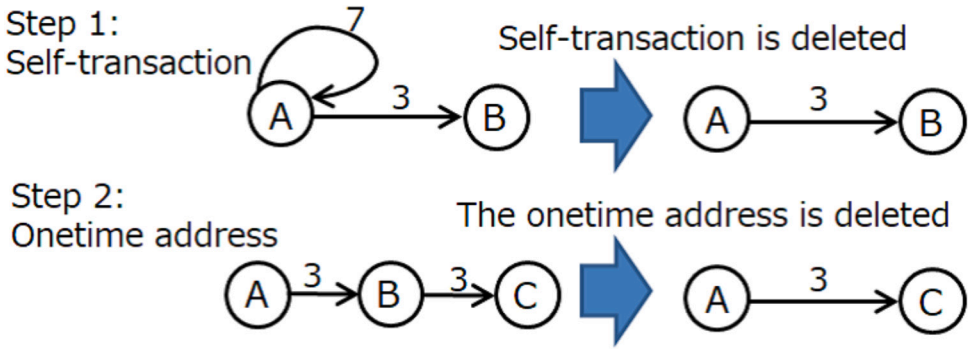

Fig. 5. Rules of UserGraph making.

3.3. Subgraph making method

In this method, the subgraph is created by searching from the target transaction. Therefore, the subgraph is called a "target rooted subgraph (TRS)". The root of a TRS is the output (destination) of the target transaction. The search creating the TRS is

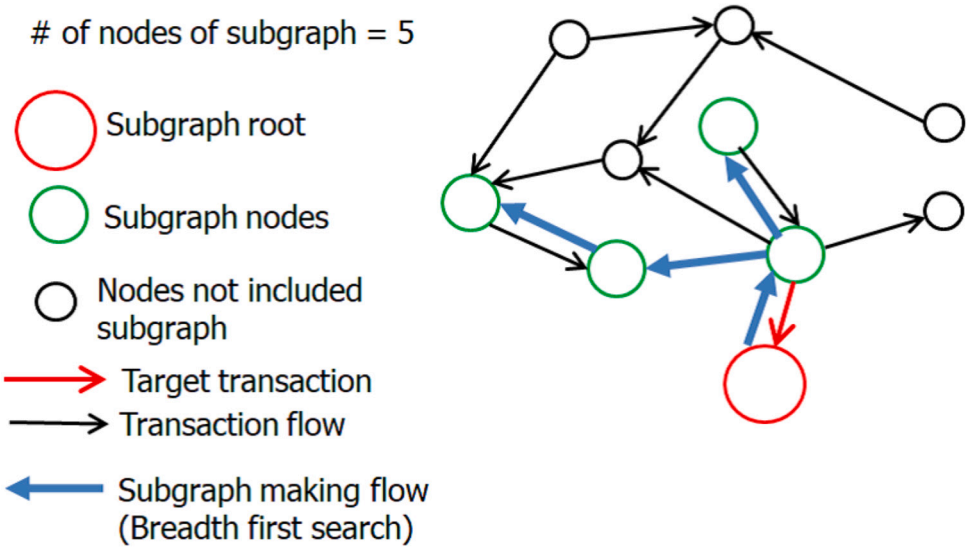

Fig. 6. Example of TRS creation.

breadth first search (BFS) from the root and the BFS is performed in the UserGraph. The number of nodes is determined by the user before creating the TRS. The BFS is performed in the direction from the destination to the source of UserGraph edges. When the number of nodes searched by the BFS is reached to the setting size of the TRS, the BFS is finished. The searched nodes become nodes of the TRS, and all the adjacent edges of the TRS nodes become edges of the TRS.

Fig. 6 illustrates an example of TRS creation. Each circle indicates a node, and each black arrow represents an edge. The number of TRS nodes is five, the red arrow is the target transaction, and the red circle is the root of the BFS. When the number of searched nodes by the BFS is five, they are considered as TRS nodes. Because the BFS searches from the output to input, the edges of the reverse direction are not searched but included in the TRS edges.

At the end of the BFS, if the number of nodes is less than the setting size of the TRS, the reverse direction edges are searched and the resulting nodes are added to the TRS. If it is still less than the setting size, randomly detected nodes in UserGraph are added to the TRS until the number of TRS nodes reaches the setting size.

The TRS has a locality related to the target transaction. Therefore, it can reduce the number of computations because the size of the TRS is smaller than that of a UserGraph. Moreover, because the local anomalies can be detected, the accuracy of anomaly detection is consequently improved.

3.4. TRS structure on GPU

The TRS and UserGraph structures are shown in Fig. 7. The TRS has three nodes and is created using five UserGraph nodes in this example. The structure of the UserGraph is based on the compressed sparse row (CSR), which is an array-based graph representation suitable for GPU processing.

CSR usually consists of two arrays: an array representing the destinations of the edges and one pointing to the edge sources. In a UserGraph, two CSRs are used for searching the deposits (InGraph) and withdraws (OutGraph). These graphs have a relation in which the directions of all edges are reversed. In a UserGraph, the edge arrays of the CSR store the transaction IDs instead of the edges' destinations to prevent duplication of the properties that correspond to the edge arrays. The properties are stored in a matrix structure, where the columns represent transaction IDs.

As mentioned in Section 3.3, the TRS nodes are detected by the BFS from the target transaction. Because the BFS applies an input search direction, InGraph is used. When BFS is performed, the searched nodes (including the root) are stored in an array. As mentioned in Section 3.3, the adjacent edges of each TRS node are the same as the adjacent edges of the same node of UserGraph. Thus, the TRS is simultaneously completed by storing the searched nodes. The TRS is the pointer of the CSR of the UserGraph, and the adjacent edges of each TRS node can be obtained across the CSR. In addition, because the TRS can access two CSRs that constitute the UserGraph, both data regarding the deposit and withdrawal can be obtained. In this method, the memory overhead is relatively low because the subgraph can be created by the addition of only one array, where the number of elements is equal to the number of nodes.

3.5. Feature extraction using TRS

As mentioned in Section 3.4, the TRS is the pointer array of the UserGraph, and it can access adjacent edges of each node using the edge arrays of the UserGraph from each pointer. Fig. 8 shows the feature extraction flow using the TRS. As an example, the

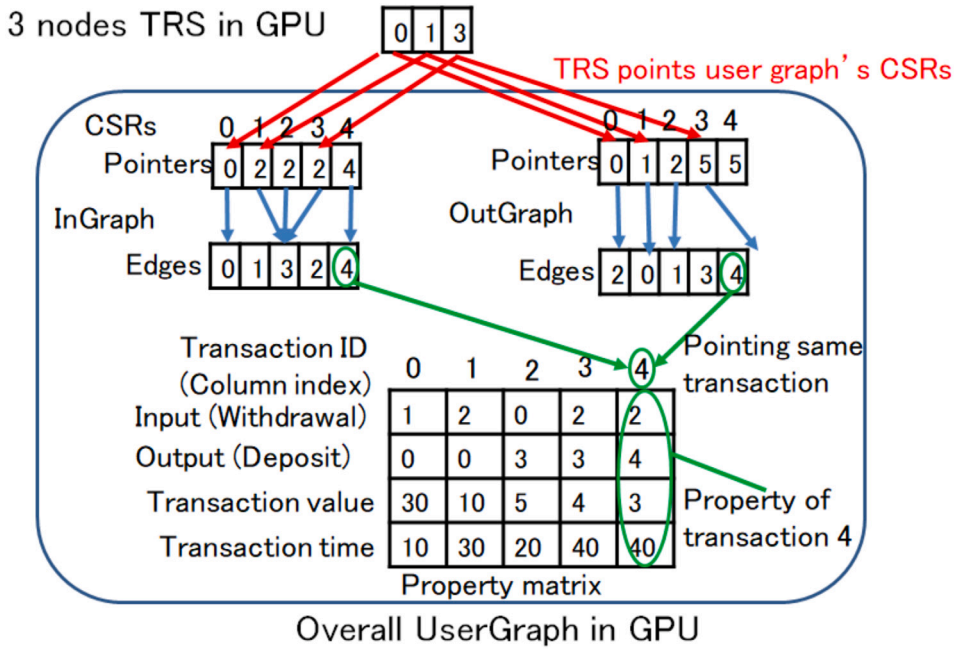

Fig. 7. Data structure of TRS and the UserGraph.

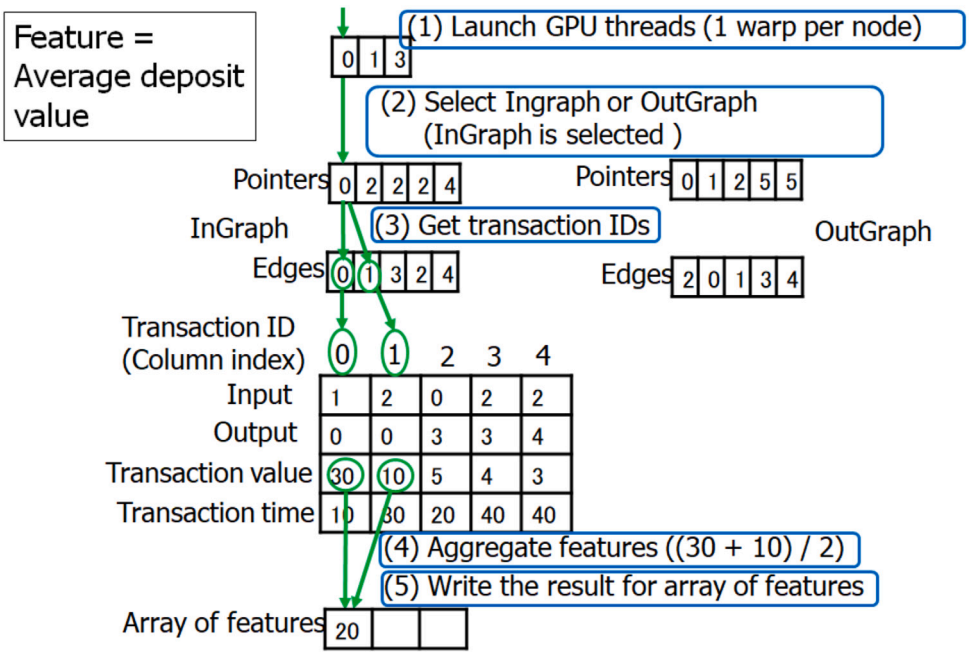

Fig. 8. Feature extraction flow using TRS.

average values of the deposits of each user are extracted from a TRS with three nodes. In a blockchain UserGraph, the degree is not uniform, and the transactions appear to be some hub users. Therefore, if the GPU threads are launched according to the number of nodes of the TRS, the hub users become a bottleneck and warp divergence occurs because the processing of hub users needs to be performed. To avoid this, the same number of warps (each warp has 32 threads) as the number of nodes of the TRS are launched, and each warp searches the elements of a user of the TRS in parallel. In this figure, the processing of a warp is shown. The flow number in the figure represents the order of processing, and each processing included the following steps:

1. GPU threads in a warp are launched per the node of TRS.
2. If the feature refers to a deposit, the InGraph is referenced; if the feature refers to a withdrawal, OutGraph is referenced. In this figure, the InGraph is referenced.
3. The transaction IDs of the adjacent edges are referenced.
4. The feature is aggregated using the transaction contents of the adjacent edges.
5. The aggregation result is stored in an array, which consists of the results of all the nodes. In this example, the array consists of three elements.

In this extraction, the number of elements in an array is equal to the number of TRS nodes. Furthermore, an array of the average deposit values is created. If multiple feature quantities are used for anomaly detection, all the features needed for anomaly detection are extracted by repeating the processing.

3.6. Anomaly detection using extracted feature quantities

The result of feature extraction is an array structure, which is suitable for GPU processing. When a GPU thread is launched for each element of the array, each thread processes an element in parallel.

The array made by feature extraction is called a "feature array". Multiple feature arrays are made because there are multiple feature types, such as the number of deposit transactions and average deposit. An anomalous transaction is detected by anomaly detection algorithms for the feature arrays. Each element of a feature array corresponds to a TRS node. A transaction is determined to be an anomaly or not depending on the corresponding detection of the node. When the target node, which is the destination of the target transaction, is detected as an anomaly, the transaction is also considered an anomaly. A K-means-based anomaly detection algorithm was implemented and evaluated in this paper. K-means is a popular clustering algorithm that can be applied to anomaly detection. The algorithm was also used in existing work [7,8], and is not novel idea of this paper. The novelty of this study is the TRS-based feature extraction that is performed before executing the algorithm. K-means detects anomalies by two thresholds after performing clustering;

1. The distance between a target node and the center of gravity of the cluster the node belongs to. The coordinate is calculated from each feature quantity.
2. The size of the cluster the node belongs to.

If (1) exceeds the threshold or (2) is smaller than the threshold, the node is considered an anomaly.

4. Extension of anomaly detection method to multiple transactions

The proposed method in Section 3 targets single transactions. However, blocks are an addition unit of transaction in blockchain, and all transactions in a block are simultaneously added. In this section, we consider the case of anomaly detection for all transactions in a block is assumed. In this case, the performance for an overall block is important. In the method presented in Section 3, the detection is performed for each transaction, and the calculation amount increases with the increase in the number of transactions in a block. In this section, we propose an extension of the aforementioned anomaly detection method using the TRS for multiple transactions.

4.1. Features of blockchain UserGraph

Transactions in blockchain are not uniform for all users but there are some transactions take place frequently for some users. UserGraph is a scale-free graph, where the degree of some of the hub nodes is large. Fig. 9 shows the degree distribution of UserGraph in the descending order by the method presented in Section 3.2 from all Bitcoin transactions up to the 397,571st block [12,13]. The average degree is 7, and the maximum degree is 3,171,134, indicating that the transactions are skewed for some users. In addition, in a blockchain system, multiple transactions of hub users are performed simultaneously and are often included in the same block. For example, the withdrawal processing of Bitmex, which is a major exchange, is performed at once per day [14].

4.2. Anomaly detection for multiple transactions

When there are multiple target transactions and the TRS includes all the targets, the TRS can perform anomaly detection for all the targets. The fact the TRS is a local graph related to the target is important for maintaining the accuracy of anomaly detection. Thus, if the TRS extends to transactions A and B, two conditions must be satisfied. In particular, the TRS should include transactions A and B, and it should be a local graph of both A and B. To meet these conditions, we propose MergedTRS. MergedTRS can be created by the following procedure:

1. Similar to the normal TRS, the TRS is made from each target transaction.
2. One step of BFS of each TRS is proceeded. One step means that all adjacent edges of frontier nodes are searched, and the new searched nodes become the new frontier nodes.

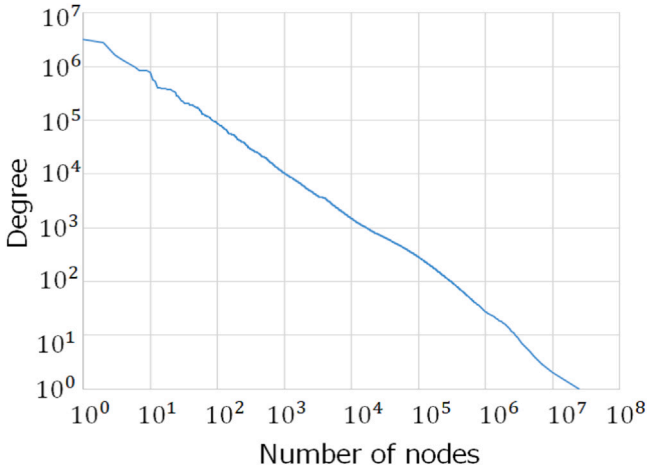

Fig. 9. Descending order degree distribution of UserGraph.

Table 1
GPU specifications used in the evaluations.

	GeForce RTX 2080 Ti
Number of cores	4352
Core clock	1350 MHz
Memory bandwidth	616.0 GB/s
Memory capacity	11 GB

3. If a node included in other TRSs is searched during the BFS, the two that include the same node are merged. The merged TRS includes the nodes of the two TRSs and the information of the frontier nodes of BFS, and the two TRSs before merging are deleted.
4. If the number of nodes of each TRS is more than the threshold after merging or BFS searching, the BFS is terminated (threshold means the setting number of nodes of TRS). Otherwise, the processing goes back to step 2 and BFS continues.

It is thus clear that a TRS is merged only when a common node is searched in the BFS during TRS making. Otherwise, the same TRS as in a single transaction is created. The number of MergedTRS nodes may overcome the threshold because the merging is performed even if the graph size of one of the merging target exceeds the threshold. However, in many cases, the merging occurs at a hub in an early stage of the search, and it is possible to create a MergedTRS including multiple target transactions. The MergeedTRS including the hub suppresses the increase in the nodes because UserGraph has hub nodes of very high degrees as mentioned in Section 4.1.

When the target transactions are all the transactions of the blockchain, the MergedTRS becomes the UserGraph, which is the same as the previous method for the overall UserGraph. This means the anomaly detection result using MergedTRS is close to that of the previous method when an extremely high number of transactions are targeted. However, when the targeted number of transactions is the same as the block size, the local graph properties of MergedTRS are maintained, as shown in the evaluation of Section 5.3.

In extreme cases, the sizes of the MergedTRSs are possible to unbalance because the size of some MergedTRSs are very large. In this case, the MergedTRSs are close to the overall user graph owing to an increased number of nodes. In the worst case, the number of nodes ($number\ of\ nodes\ of\ TRS \times number\ of\ targets - 1$) and the size of MergedTRS are significantly larger than those of the original TRS. However, in any case, the size of the MergedTRS cannot exceed the size of the overall graph, and the latency of the anomaly detection in the MergedTRS is less than or equal to the latency in the overall graph.

5. Evaluations

5.1. Evaluation environment

In this evaluation, an Intel Xeon E5-2637v3 running at 3.5 GHz was used for CPU processing and NVIDIA GeForce RTX 2080 Ti (Table 1) with a compute unified device architecture (CUDA) version 7.5 was used for GPU processing. The data of actual Bitcoin transactions up to the 397,571st block (February 2016) were used for the evaluation [12,13]. When the size of the graph was varied, the part of the graph was first deleted from the latest transactions. The Kmeans-based algorithm was used for anomaly detection. The number of clusters was 16 in the TRS-based method and 18 in the UserGraph-based methods [7,8]. The first threshold of the algorithm was 0.5x the minimum distance of all gravity center pairs. The second threshold of the algorithm was equal to cluster size, specifically 1/100 the number of nodes of the TRS in the TRS-based method or 1/500 in the UserGraph in the UserGraph-based method. These thresholds were determined such that all large anomalous transactions (transfer value > 500 Bitcoin) could be

Table 2
Comparison of original data and proposed UserGraph.

	# of users	# of transactions
Original data	124,857,865	322,855,563
After applying proposed method	41,889,542	291,149,791

Table 3
TP rate and FP rate of UserGraph method [7].

10 to 50	50 to 100	100 to 500	over 500	FP
0.01	0.01	0.27	1.00	0.02

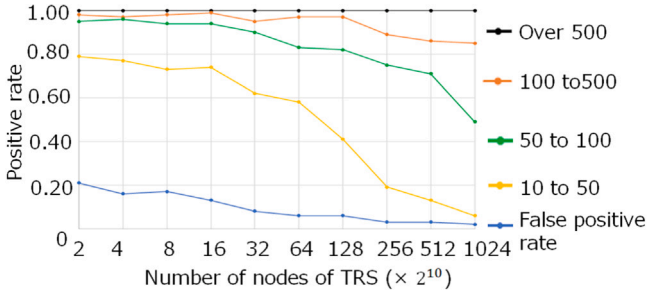

Fig. 10. TP rate of anomalous transactions and the FP rate of normal transactions.

detected in each method. Four features were used: average deposit (input), average withdrawal (output), the number of deposits, and the number of withdrawals.

Table 2 compares the original data and proposed UserGraph. The number of users of the original data represents the number of addresses, and the number of transactions of the original data indicates the number of transaction outputs. As can be seen, by using the proposed method, the number of addresses by 66% could be reduced. The UserGraph was used for the evaluation.

5.2. Accuracy comparison between UserGraph and TRS-based method

In this section, the accuracy of anomaly detection is compared between the UserGraph-based (existing) and TRS-based (proposed) method. The accuracy evaluation is performed in GPU because the detection results of each method between CPU and GPU processing are equivalent. The difference in the results between CPU and GPU processing is execution time and the time is shown in Section 5.4.

Notably, anomalous and normal transactions are required for accuracy evaluation. An anomalous transaction is a fraudulent transaction in blockchain. However, the actual fraudulent transaction information is not disclosed. Therefore, in the evaluation, artificial anomalous transactions were inserted to the actual blockchain transaction data. The transactions were created by the following steps:

1. A user who has a balance in the setting range was selected randomly.
2. All deposits of the user were transferred to a new user.

The scale of fraudulent transaction was set by the range of step (1). The normal transaction was selected randomly from all the transactions in the UserGraph. The number of executions was 100 for each type. The true positive (TP) rate and the false positive (FP) rate the in previous method are presented in Table 3. The unit of fraudulent transaction size is BTC (Bitcoin). The threshold was determined such that the TP rate in large transactions (>500 BTC) was 1.00 and the FP rate was as low as possible (specific number is described in Section 5.1). The TP rate decreases with the decrease in scale; this is because the scale of anomaly is too small for the size of overall Bitcoin network. At <100 BTC scales, the method can barely detect anomalies.

Fig. 10 depicts the TP and FP rates under the same conditions as the previous method when the number of TRS nodes is varied from 2×2^{10} to 1024×2^{10}.

Upon comparing the TRS and the previous method, it was found that the TRS can detect small-scale anomalies, which is not possible with the previous method. Observing the differences between the sizes of the TRS, it can be concluded that the small-scale anomaly (especially 10 to 50) detection rate decreases with increase in the size of the TRS because there are fewer characteristics of the local graph in a large TRS. The FP rate also decreases with size. When the size of TRS is 1024×2^{10}, the FP rate is almost the same as that of the previous method (both FP rates are 0.02), and the TP rate at a small scale is higher than that of the previous method. Thus, the proposed method is more accurate than the previous method.

It should be noted that the size of the TRS should be selected depending on the size of the targeted anomalies. In this evaluation, we mainly considered the 128×2^{10} case, which can detect anomalies of >100 BTC with an accuracy of over 90%.

Table 4
Accuracy evaluation in multiple transaction cases.

# of targets	10–50	50–100	100–500	500+	FP	Max size
100	0.49	0.84	0.96	1.00	0.04	196,882
200	0.41	0.64	0.75	0.92	0.03	207,472
300	0.47	0.82	0.82	0.99	0.03	205,631
400	0.43	0.82	0.97	1.00	0.03	221,865
500	0.58	0.73	0.86	0.97	0.03	218,801

Table 5
Accuracy evaluation in multiple transaction cases for low threshold.

# of targets	10–50	50–100	100–500	500+	FP
100	0.66	0.85	0.98	1.00	0.05
200	0.48	0.70	0.84	1.00	0.04
300	0.53	0.82	0.85	1.00	0.04
400	0.53	0.83	0.98	1.00	0.04
500	0.58	0.75	0.89	1.00	0.04

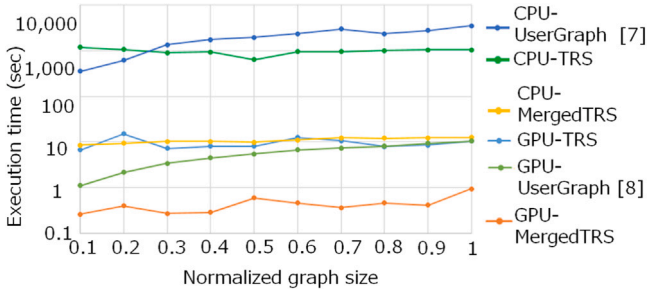

Fig. 11. Comparison of execution times of anomaly detections.

5.3. Accuracy evaluation in multiple transaction targeted case

Table 4 shows the maximum number of nodes and accuracy of the anomaly detection by the MergedTRS method when the number of targeted transactions is varied from 100 to 500. The number of nodes is set to 128×2^{10}. The number of targeted transactions was determined to exceed the average number of transactions of one block of Ethereum (68 as of March 2020 [15]). The target transaction set consists of ten anomalous and multiple normal transactions, and ten sets of evaluation are performed. The maximum size of the MergedTRS is only 1.5-1.7x larger than the setting size although the number of targeted transactions ranges from 100 to 500. This shows that the characteristics of the local graph can be maintained when the targeted number of transactions is comparable to the block size. Thus, the accuracy tends to be the same as that of the TRS, whose number of nodes is 128×2^{10}. However, the TP and FP rates decrease, implying that a lower threshold is required for the MergedTRS method if the same positive rates as those of the TRS are required.

Table 5 shows the accuracy of anomaly detection of the MergedTRS method in cases when the threshold is low. The thresholds were determined such that the size of the cluster including the target is 1/50 the MergedTRS size and 0.4x the difference of the minimum distance of all pairs of center of gravity. Furthermore, it was ensured that the TP rate is 1 for large anomalous transactions. All positive rates in Table 5 increase as compared to those presented in Table 4. As a result, the TP and FP rates are almost in the range of the TP and FP of the TRS, whose sizes range from 128×2^{10} to 256×2^{10}. From these results, it can be concluded that the MergedTRS is an extension of the TRS-based method and can be applied to multiple transactions without compromising accuracy.

5.4. Comparison of execution time of anomaly detections

The execution time of anomaly detections for 100 transactions was evaluated. The execution time includes the time required for creating the subgraph, extraction of features, and execution of K-means for anomaly detection. Fig. 11 compares the following methods: (1) CPU-based UserGraph (CPU-UserGraph [7]), (2) GPU-based UserGraph (GPU-UserGraph [8]), (3) CPU-based not merged TRS (CPU-TRS), (4) GPU-based not merged TRS (GPU-TRS), (5) CPU-based MergedTRS (CPU-MergedTRS), and (6) GPU-based MergedTRS (GPU-MergedTRS).

The graph size shown on the x-axis indicates the number of transactions normalized by the graph size described in Table 2. The previous and Merged-TRS methods can process 100 transactions simultaneously, whereas the TRS methods can perform 100 transactions sequentially. Thus, the TRS times represent the execution time for 100 runs. The number of TRS and MergedTRS setting nodes is 128×2^{10}.

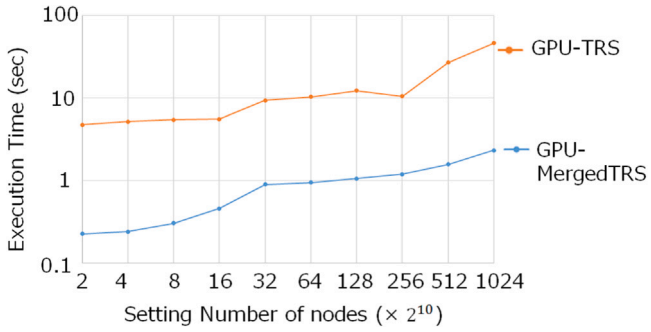

Fig. 12. Change in the execution time of the anomaly detections over the setting number of nodes.

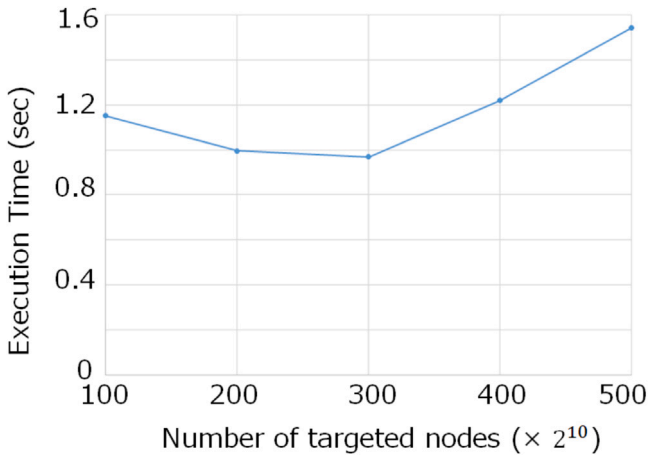

Fig. 13. Effect of number of targeted transactions on execution time.

In the UserGraph methods, when the number of nodes of the UserGraph is N, the amount of computation is O(N). Therefore, the execution time increases with the number of transactions. On the contrary, the execution time of the TRS-based methods is almost constant regardless of the variations in the size of the UserGraph. This result therefore reveals that the TRS has high scalability. However, the GPU-TRS method is not faster than the GPU-UserGraph because it is executed 100 times. If the target number of transactions is 1, the GPU-TRS is 83.2x faster than the GPU-UserGraph, implying that the latency of the GPU-TRS is much better than that of the GPU-UserGraph. Moreover, the MergedTRS can accelerate the TRS-based method by avoiding the repetition when multiple transactions are targeted. As a result, in the case of the maximum graph size, the GPU-MergedTRS is 11.1x faster than the GPU-UserGraph and 13.3x faster than the GPU-TRS.

5.5. Effect of number of TRS nodes or targeted transactions changed on execution time

Fig. 12 illustrates the effect of changing the setting number of nodes on the execution time of anomaly detection by the GPU-MergedTRS and GPU-TRS. When the number of TRS nodes is small, GPU parallelism is not fully exploited. Therefore, the increase of execution time is less than the increase of the number of TRS nodes in both methods.

Fig. 13 shows the execution time when the number of targeted transactions is changed. The setting number of nodes is 128×2^{10}. The amount of computation increases with the number of targeted transactions because the overhead of creating the MergedTRS and the maximum number of nodes increase. However, until the number of targeted transactions is 300, the execution time decreases because the number of not-merged TRSs decreases with the increase in the number of the targeted transactions. The execution time is suppressed by 1.5 times, even when the number of targeted transactions is changed from 100 to 500, because the rate of increase of the overhead and graph size is smaller than that of the number of targeted transactions.

6. Applications of proposed method

As shown in the evaluation, the proposed method can detect anomalies in less than a second, which is shorter than the approval interval of blockchain. The main application of the proposed method is in prevention of theft, which was mentioned as a problem in blockchain. In particular, because most large-scale thefts in blockchain are caused during exchanges of cryptocurrencies [16],

we consider anti-theft at the exchanges as an application of the proposed method. Anti-theft is performed by overwriting abnormal transactions before the approvals [17]. If a private key is stolen, illegal transactions can be issued repeatedly. Therefore, it is not sufficient to simply detect and correct anomalies. It is necessary to operate the method in combination with a mechanism for transferring assets to an address without a stolen secret key.

In addition, alerts of abnormal transactions are also assumed as an application. If a secret key of an address is stolen, the transferred asset to the address after the theft is also stolen. Users can prevent secondary damage by canceling the transfer to the address that has been detected as abnormal.

7. Conclusions

In a blockchain system, it is necessary to detect anomalous transactions at high speed because transactions cannot be modified after approval. However, existing anomaly detection methods do not meet this requirement, because they require the overall transaction data, and the computational cost increases with the number of transactions. This problem could be avoided by the use of the TRS-based method, in which the BFS is performed from a target transaction; TRS is a local graph of the target, and the method using the TRS is scalable in terms of the latency when the number of overall blockchain transactions increases. Using real Bitcoin data, the execution time of the TRS-based method was constant, and the latency of the TRS method using GPU was 83.2 times faster than an existing method using GPU. However, the method is not suitable for multiple targets because the method targets only one vertex at a time. Therefore, TRS was extended to MergedTRS, which is an extension of TRS to target multiple transactions. MergedTRS is created by merging multiple TRSs to use their common vertices. In the evaluation, the size of MergedTRS was a maximum of 2.2x that of TRS despite the number of targets ranging from 100 to 500. The execution time of the MergedTRS method using GPU was 11.1x faster than the existing method using GPU when the target number of transactions was 100. The execution time was shorter than two second, which is sufficiently below the approval time of blockchain systems. In terms of detection accuracy, the TRS-based method could improve the TP rate in small-scale transactions because the subgraph has locality that is suitable for local anomaly detections. Moreover, the TP rate in large-scale transactions and FP rate were close to those of the previous method.

CRediT authorship contribution statement

Shin Morishima: Conceptualization, Methodology, Software, Validation, Investigation, Data curation, Writing, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

The author expresses his gratitude for the assistance from to JSPS KAKENHI Grant Number JP19K24350.

References

[1] Nakamoto S. Bitcoin: A Peer-to-Peer electronic cash system. 2020, https://www.bitcoin.com/bitcoin.pdf [Accessed 7 Aug 2020].
[2] Counterparty. 2020, https://counterparty.io/ [Accessed 7 Aug 2020].
[3] Nordrum A. Wall street occupies the Blockchain - financial firms plan to move Trillions in assets to Blockchains in 2018. IEEE Spectr 2017;40–5.
[4] Ethereum project. 2020, https://www.ethereum.org/ [Accessed 7 Aug 2020].
[5] Luu L, Chu D-H, Olickel H, Saxena P, Hobor A. Making smart contracts smarter. In: Proceedings of the ACM SIGSAC conference on computer and communications security. 2016, p. 254–69.
[6] Breunig MM, Kriegel H-P, Ng RT, Sander J. LOF: Identifying density-based local outliers. In: Proceedings of the 2000 ACM SIGMOD international conference on management of data. 2000, p. 93–104.
[7] Pham T, Lee S. Anomaly detection in bitcoin network using unsupervised learning methods. Comput Res Repos 2016;1–5, abs/1611.03941.
[8] Morishima S, Matsutani H. Acceleration of anomaly detection in blockchain using in-gpu cache. In: Proceedings of the 16th IEEE international symposium on parallel and distributed processing with applications (ISPA'18). 2018, p. 244–51.
[9] Anh DTT, Ji W, Gang C, Rui L, Chin OB, Kian-Lee T. BLOCKBENCH: A framework for analyzing private Blockchains. In: Proceedings of the international conference on management of data. 2017, p. 1085–100.
[10] Signorini M, Pontecorvi M, Kanoun W, Pietro RD. BAD: Blockchain anomaly detection. 2018, p. 1–8, CoRR abs/1807.03833.
[11] Gaihre A, Luo Y, Liu H. Do bitcoin users really care about anonymity? An analysis of the Bitcoin transaction graph. In: Proceedings of the international conference on big data. 2018, p. 1198–207.
[12] Bitcoin network dataset. 2020, https://senseable2015-6.mit.edu/bitcoin/ [Accessed 7 Aug 2020].
[13] Kondor D, Posfai M, Csabai I, Vattay G. Do the rich get richer? An empirical analysis of the bitcoin transaction network. PLoS One 2014;9(2):1–10.
[14] Bitmex FAQ. 2020, https://www.bitmex.com/app/faq#When-are-Bitcoin-withdrawals-processed [Accessed 7 Aug 2020].
[15] Etherscan. 2020, https://etherscan.io/ [Accessed 7 Aug 2020].
[16] Chohan UW. The problems of cryptocurrency thefts and exchange shutdowns. 2020, Available at Social Science Research Network (SSRN) https://ssrn.com/abstract=3131702 [Accessed 7 Aug 2020].
[17] Antonopoulos AM. Mastering bitcoin. 2nd ed.. O'Reilly Media; 2017.

Shin Morishima He is an assistant professor in the Faculty of Engineering at Toyama Prefectural University. His research interests include Blockchain and GPU-based systems. He received a PhD in engineering from Keio University in 2018.