IEEE Open Journal of the Computer Society

Received 25 March 2020; accepted 16 April 2020. Date of publication 5 June 2020;
date of current version 3 July 2020. The review of this paper was arranged by Associate Editor Zhou Su.

Digital Object Identifier 10.1109/OJCS.2020.2990458

XBlock-ETH: Extracting and Exploring Blockchain Data From Ethereum

PEILIN ZHENG [1], ZIBIN ZHENG [1,2] (Senior Member, IEEE), JIAJING WU [1] (Member, IEEE), AND HONG-NING DAI [3] (Senior Member, IEEE)

[1]Sun Yat-sen University, Guangzhou 510275, China
[2]National Engineering Research Center of Digital Life, Sun Yat-sen University, Guangzhou 510275, China
[3]Macau University of Science and Technology, Macau SAR 999078, China

CORRESPONDING AUTHOR: ZIBIN ZHENG (e-mail: zhzibin@mail.sysu.edu.cn)

This work was supported in part by the National Key Research and Development Program under Grant 2016YFB1000101, in part by the National Natural Science Foundation of China under Grants 61973325, in part by the Fundamental Research Funds for the Central Universities under Grant 17lgpy120, in part by Key-Area Research and Development Program of Guangdong Province under Grant 2019B020214006, and in part by Macao Science and Technology Development Fund under Macao Funding Scheme for Key R & D Projects under Grant 0025/2019/AKP.

ABSTRACT Blockchain-based cryptocurrencies have received extensive attention recently. Massive data has been stored on permission-less blockchains. The analysis of massive blockchain data can bring huge business values. However, the absence of well-processed up-to-date blockchain datasets impedes big data analytics of blockchain data. To fill this gap, we collect and process the up-to-date on-chain data from Ethereum, which is one of the most popular permission-less blockchains. We name such well-processed Ethereum data as XBlock-ETH, which consists of transactions, smart contracts, and cryptocurrencies (i.e., tokens). However, it is non-trivial to partition and categorize the collected raw Ethereum data to the well-processed datasets since the whole processing procedure requires sophisticated knowledge on software engineering as well as big data analytics. Moreover, we also present basic statistics and exploration for each of the well-processed datasets. Furthermore, we also outline the possible research opportunities based on XBlock-ETH, with the data and code released online.

INDEX TERMS Blockchain, data analytics, ethereum, smart contracts.

I. INTRODUCTION

Blockchain has attracted extensive attention from both academia and industry in recent years. Among the diverse blockchain systems, substantial efforts have been made on the permissionless blockchain (or public blockchain) due to its decentralization [1]. The idea of permissionless blockchain was firstly proposed and implemented on Bitcoin [2]. In a blockchain system, each peer holds a ledger being considered as a public tally that is essentially temper-resistant. Ethereum [3] is another most popular permission-less blockchain system that enables Turing-complete smart contracts. The proliferation of blockchain systems has lead to the generation of the massive amount of blockchain data. Take Bitcoin as an example. There are nearly 242 GB Bitcoin data by the third quarter of 2019, as reported by Statista (https://www.statista.com/). In this paper, we focus on the data of Ethereum rather than Bitcoin, since Ethereum provides richer data types. For another example, more than 16,000,000 smart contracts are deployed on Ethereum. As the Ethereum community has published two token protocol to enable easier Initial Coin Offerings (so-called ICO) for users [4], over 100,000 kinds of ERC20 token and 1,600 kinds of ERC721 token are available to be transferred on Ethereum where ERC stands for Ethereum Request for Comments.

The massive blockchain data provides researchers with both huge business values and great opportunities [5] due to openness, decentralization, and temper-resistance of blockchain systems. Take business trading data as an example. In the past, it is difficult for researchers to obtain the real business trading data because of the privacy or ownership concerns of data owners. However, all the data in incumbent blockchain systems are all publicly available. Meanwhile, the blockchain data in permission-less blockchains can be accessed almost everywhere due to the

decentralization of blockchain systems. Moreover, the distributed consensus of blockchains also guarantees the temper-resistance of blockchain data. In addition to blockchain transactions, Ethereum (or its alternatives) also consists of both smart contracts and cryptocurrencies. Big data analytics of blockchain data can advance the developments in fraud detection of transactions, vulnerability detection of smart contracts and software development of smart contracts, etc.

However, there are a number of challenges in big data analytics of blockchain data, especially in Ethereum: (1) Difficulty in data synchronization at Blockchain peer. Due to the bulky size of the blockchain, it takes a long period to fully synchronize entire blockchain data at a node (i.e., a peer) newly connected with the blockchain. For example, it takes more than one week and over 500 GB storage space to fully synchronize the entire Ethereum at a peer. The high expenditure of massive storage space and network bandwidth due to blockchain data synchronization impedes the analysis of blockchain data. (2) Challenge in blockchain data extraction and processing. Blockchain data is stored at clients in heterogeneous and complex data structures, which cannot be directly analyzed. Meanwhile, the underlying blockchain data is either binary or encrypted. Thus, it is a necessity to extract and process binary and encrypted blockchain data so as to obtain valuable information. However, it is non-trivial to process heterogeneous blockchain data since conventional data analytic methods may not work for this type of data. (3) Absence of general data extract tools for blockchains. Although many studies provide open-source data extraction tools of blockchain data, most of them can only support to extract partial blockchain data (not all the data). Moreover, most of the existing tools can only fulfill specific research tasks. (4) Absence of basic data explorations for blockchains. Existing studies only focus on specific data analysis of blockchain data, e.g., transaction graph [6], contract security [7]. However, basic data explorations like statistic analysis, text analysis, and data visualization are missing in most of the existing tools.

To address the above challenges, we propose a blockchain data analytics framework, namely eXplore Blockchain ETH (XBlock-ETH), to analyze Ethereum data. In particular, we extract the raw data consisting of 8,100,000 blocks from Ethereum. The raw data includes three types of Ethereum data: *blocks*, *traces*, and *receipts*. Since the analysis on the raw Ethereum data is difficult, we partition and categorize the obtained Ethereum Blockchain data into six datasets: *(1) Block and Transaction*, *(2) Internal Ether Transaction*, *(3) Contract Information*, *(4) Contract Calls*, *(5) ERC20 Token Transactions*, and *(6) ERC721 Token Transactions*. The new categorization of Ethereum data can help other researchers to explore and analyze Ethereum data in a more convenient way. However, it is non-trivial to partition and process the raw data since it requires substantial efforts in extracting metadata information from raw data and associating with six datasets. We then conduct a statistic analysis of the six refined datasets. We also discuss the potential applications of XBlock-ETH, such as blockchain system analysis, smart contract analysis, and cryptocurrency analysis.

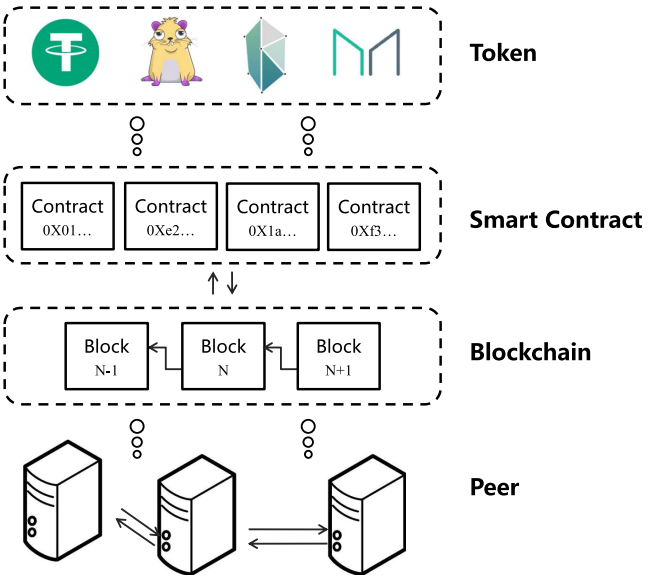


FIGURE 1. Overview of Ethereum Blockchain.

In summary, we highlight the major contributions of this paper as follows:

- The XBlock-ETH data contain the comprehensive on-chain data in contrast to previous works (only cover partial Ethereum data). In particular, it includes blockchain data, smart contract data, and cryptocurrency data. Moreover, these well-processed datasets can be easily used for data exploration. Furthermore, XBlock-ETH data formally released online[1] has been periodically updated.
- The XBlock-ETH framework also offers basic statistics and exploration functions to analyze blockchain datasets. This paper also outlines the research opportunities brought by XBlock-ETH. In particular, we discuss the applications of XBlock-ETH in aspects of blockchain system analysis, smart contract analysis, and cryptocurrency analysis.

The rest of this paper is organized as follows. Section II first gives an overview of blockchain and smart contract technologies. Sections III, IV then present raw data acquisition from Ethereum and data exploration of six datasets. Section V discusses the applications of XBlock-ETH data. Section VI surveys related work. Finally, the paper is concluded in Section VII.

II. BACKGROUND

Fig. 1 presents an overview of Ethereum blockchain, which consists of the following layers from bottom to top: peer, blockchain, smart contract, and token. We next review the basic concepts of each layer in Ethereum.

[1] http://xblock.pro/on-chain-data/

A. PEER AND BLOCKCHAIN

In a nutshell, a blockchain is essentially a chain-like data structure consisting of a number of consecutively-connected blocks. The chain has been maintained by all the peers in a peer-to-peer (P2P) blockchain network. In a certain period of time, only one block can be confirmed by the entire blockchain network through a consensus protocol. The block containing the confirmed transactions at that time and the hash value of the previous block have been generated by a *peer* (a.k.a. miner). After being generated, the block will be validated independently by the other peers. Once the block is validated and confirmed by most of the peers in the blockchain network, the transactions in the block will be considered as *completed*. In this way, each peer can trust the whole blockchain (a.k.a. ledger) since the transactions have been validated by all the peers. In other words, blockchain enhances the trustworthiness of transaction data through duplicating computation and storage at all peers.

Thanks to the completeness of blockchain data in each permission-less blockchain peer, researchers can obtain the entire blockchain data via connecting a blockchain peer to the blockchain network. The blockchain data that consists of all the operations done by the users and miners in the blockchain contains substantial business values. For example, the transaction records are essentially operations done by different business parties. The analysis of the blockchain data can help to understand user behaviors in a real-world economic system (e.g., money transferring). Meanwhile, there is a rapid growth of blockchain data, especially in Bitcoin and Ethereum, with the proliferation of blockchain users and transactions. The analysis of blockchain data can also be beneficial to predict the economic trend.

B. SMART CONTRACT

The smart contract that was proposed even earlier than blockchain [8] is a promising technology to reshape the modern industry. Blockchain-based smart contracts are essentially computer programs, in which the execution states are stored on top of the blockchain. The blockchain transactions are the messages representing the deployment or invocations of smart contracts. Therefore, blockchain guarantees the trustworthiness of smart contracts.

The incumbent blockchain systems have enabled smart contracts. For example, Bitcoin enables users to run a simple script program during the execution of transactions. This script can be regarded as a simple blockchain-based smart contract. However, the Bitcoin script is not Turing-complete so that it cannot enables complex logic expressions in the contract. In contrast, Ethereum enables Turing-complete smart contracts. In Ethereum, a smart contract is executed in an environment called Ethereum Virtual Machine (EVM). EVM reads and writes the states (stored in the key-value pair) as the actions defined in a smart contract. During the contract execution, a miner uses "*Gas*" as a unit to evaluate the consumption of one smart contract. During the contract execution, the contract user is charged by the "`GasUsed`" and "`GasPrice`". The more "`GasPrice`" that the users promise to pay for the miner, the faster the contract executes. After the transactions (i.e., operations) are done, EVM will generate a hash value of the state and record it into the blockchain. Therefore, we can learn from Fig. 1 that smart contracts on Ethereum are not directly stored on the blockchain. They are essentially stored in the states that have been operated by the blockchain.

C. TOKENS AND CLIENTS

It is worth mentioning that Ethereum has two standard token protocols (a.k.a. templates) of smart contracts. These token protocols define the standard variables, functions, and interfaces in the smart contract. With the protocols, users can issue tokens (or so-called cryptocurrencies) based on smart contracts on top of Ethereum. There are four typical tokens USDT,[2] Cryptokitties [9], Kyber [10], and MarkerDAO[3] as shown in Fig. 1 (i.e., the top layer). For example, a user can publish an ERC20 contract on Ethereum issuing tokens to others. After that, any other users (even contracts) can receive or send the token without a centralized authority (e.g., stock exchange). The standard token protocols greatly enrich the ecosystem of Ethereum so as to make Ethereum become a flexible financial system. In Section IV-E and IV-F, we will explore the data of tokens in Ethereum.

Ethereum allows that any computer programs can join into the network if they meet the requirement of the protocol just like P2P protocols (e.g., BitTorrent). As a result, there are a number of diverse Ethereum clients that can validate the blocks and transactions. Among most of Ethereum clients, Go-Ethereum (Geth) and Parity have been the most widely used according to the statistic from Ether nodes.[4] Both of them provide JSON-RPC interfaces for users to interact with Ethereum blockchain. Through the JSON-RPC interfaces, the user can obtain the blockchain data from Ethereum. Geth has been generally used in many previous studies, while the interfaces designed in Geth are not suitable for data acquisition. Even though many researchers attempted to modify source codes of Geth to obtain the detailed run-time data, the whole procedure of the code modification is time-consuming and complex. In addition, the obtained data is not absolutely accurate in some cases. Different from Geth, Parity better designs the interfaces so that it can obtain the index of each block corresponding to each piece of the data that we need. The details on data acquisition of blockchain data will be described in Section III.

III. RAW DATA EXTRACTION FROM ETHEREUM

This section describes the procedure of how the raw data was obtained from Ethereum blockchain. Fig. 2 illustrates the typical Ethereum transaction execution flow from Block *N*

[2]https://tether.to/
[3]https://makerdao.com/
[4]https://ethernodes.org

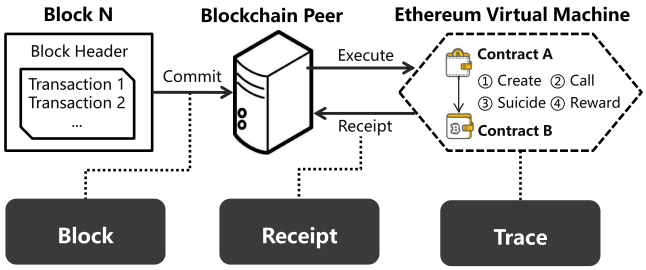


FIGURE 2. Raw data collection during Ethereum transaction flow.

to EVM through Blockchain peer. During this procedure, we collect the three types of blockchain raw data: Block, Receipt, and Trace. We next describe the details on the composition and acquisition of each kind of raw data.

A. BLOCK

Block data is directly stored in the Ethereum blockchain. Each block consists of two components:

- **Block Header:** Block header is the basic information of a block, including the miner's address, timestamp, gas limit, etc.
- **Block Transactions:** Block transactions construct the body of the block. Each transaction consists of the fields: *From*, *To*, *Value*, *Input*, etc. If the transaction is used to deploy a contract, the *To* field is "`null`" in the block transaction.

Almost all the Ethereum clients, including Geth and Parity, offer the interfaces to query the blocks. For example, "`eth_getBlock`" is available in both Geth and Parity with similar efficiency. However, we can only obtain little information about blockchain users by analyzing the block data. This is because the input of block transactions only represents operations to EVM in the contract deployment phase, while the contract code will be stored only at the end of the transaction execution, and it is not the same as the input of the transaction. Thus, we cannot obtain the exact contract code in the block transaction. Meanwhile, in the contract invocation phase, we cannot know whether the transaction is executed successfully or what kinds of errors thrown during the transaction execution since sometimes a contract will send messages or cryptocurrencies to other contracts.

B. TRACE

Trace data is essentially the detailed run-time data that was generated in EVM (e.g., internal contract calls, transferring money from the contract to a person). Trace data cannot be directly obtained or observed from the block data but can be recorded during the contract execution. In this paper, trace data is referred to the data that cannot be obtained before or after the transaction execution, but only appears during the execution. Trace data includes the following types:

- **Create** is the trace, including the creator, code, and initial balance when a smart contract is deployed. The creator of a contract can be a person or another smart contract.
- **Call** occurs when money or messages are transferred through different Ethereum addresses. Contract call or Ether transferring is shown as a "`Call`" trace.
- **Suicide** is the trace that smart contract "`suicide`" deletes its code, and refunds the value to a specific account.
- **Reward** is the trace that miners get the Ether reward when they mine a block. The reward value varies depending on the contribution of the miners.

In Geth, the interface of trace is "`debug_traceTransaction`". However, this interface returns all the operations during the transaction, resulting in large resource consumption and low efficiency. Thus, many previous studies attempt to modify the source codes of Geth to obtain detailed run-time data, while this procedure is extremely time-consuming. In this paper, we adopt "`parity_trace`" in Parity to obtain the trace data. This interface is provided and maintained by the official developers so that the correctness is guaranteed in contrast to Geth. Meanwhile, it also provides enough information that we need, such as the basic trace types and errors. Moreover, another advantage of Parity is the updating convenience as the data is indexed by blocks.

C. RECEIPT

After the transaction is executed, some of the Ethereum states have been changed (e.g., the balance of the account in a token contract). Then the clients need to know what has been changed. To reduce the query overhead of clients, many contracts leave one kind of outputs called "`Event`" in the execution. For example, a standard token contract will output a "`Transfer(from,to,value)`" event to let the clients know what happens during the execution. This kind of output is a one-way output, as it is just written in the receipt of the transaction, and can be read by external clients or persons but cannot be read by internal EVMs.

Section IV will then give the statistics of Ethereum data. In particular, there are over 100,000 kinds of cryptocurrencies using smart contracts on Ethereum. As for these token contracts, the receipt data is an important source to learn about the holders, owners, and user behaviors. Thus, it is necessary to obtain receipt data. Both Geth and Parity provide the interfaces to get the transaction receipts. The main difference between Geth and Parity interfaces lies in the query index of the receipts. In particular, the receipt of the interface of Geth is "`eth_getTransactionReceipt`" that is indexed by the transaction hash, while the interface of Parity is "`parity_getBlockReceipts`" that is indexed by block number. In this way, Parity is much more efficient than Geth since it can return a batch of receipts in one query.

In summary, there are three kinds of raw datasets that can be obtained in Ethereum: block, trace, and receipt. More specifically, it takes almost 2 weeks to synchronize the entire Ethereum blockchain and obtain the raw data, with the

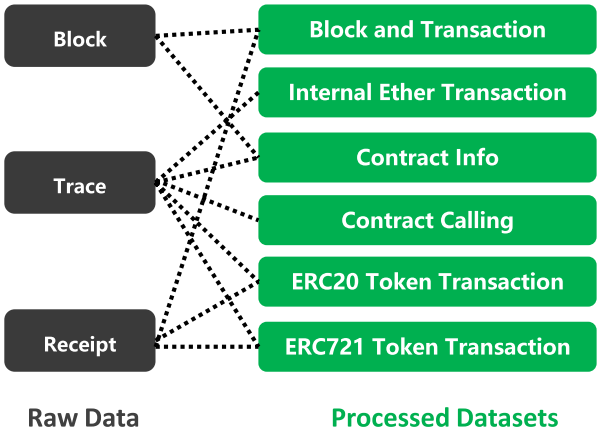


FIGURE 3. Mapping from raw data to datasets.

implement in Shell, NodeJS, and Python. After compression, the size of the data is about 313 GBytes. Because of the massive volume and redundant information of the raw data, data procession is necessary to simplify data representation and fasten data analysis for further study.

IV. DATA EXPLORATION OF ETHEREUM

In this section, we process the obtained raw data from Ethereum and divide it into six datasets: (1) Block and Transaction, (2) Internal Ether Transaction, (3) Contract Info, (4) Contract Call, (5) ERC20 Token Transaction, and (6) ERC721 Token Transaction. The relationship from the raw data to the processed datasets is shown in Fig. 3. The reason for dividing these six datasets is that we want to find the minimum necessary subset of data for the researchers in a specific research field. For example, if a researcher wants to study the Ether transferring network, he can only study the Dataset 2 (Internal Ether Transaction), rather than dealing with the raw data or other sub-datasets, saving his workload.

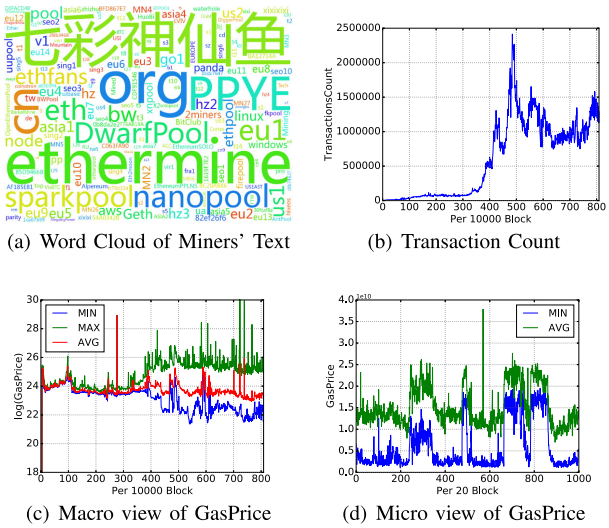
This section will introduce how the datasets are generated, with statistics and observations.

A. DATASET 1: BLOCK AND TRANSACTION

To investigate the basic statistics of Ethereum, we extract the information about the blocks and the transactions inside the blocks. In particular, there are 8,100,000 blocks and 491,562,222 transactions generated from the block data. For each block, we also obtain the statistic values of the "`gasPrice`": minimum, average, and maximum. Meanwhile, corresponding to the hash of each transaction, the fields of "`minerReward`," "`gasUsed`" and "`error`" are extracted from the receipt and trace. Regarding the miners of the Ethereum blockchain, there are 5,122 unique addresses of miners, as shown in Table 1. It implies that there are no more than 5,122 peers that serve as miners since one peer may own more than one address. Meanwhile, each miner has the right to write extra texts in the block. So, we also

TABLE 1. Statistics of Dataset 1

Statistics	Values
No. of Blocks	8,100,000
No. of Transactions	491,562,222
No. of Miner Addresses	5,122
Mean of Transaction Counts per Block	60.68
Mean of Block Time	15.33 seconds
Mean of Block Size	11,457 bytes



(a) Word Cloud of Miners' Text (b) Transaction Count

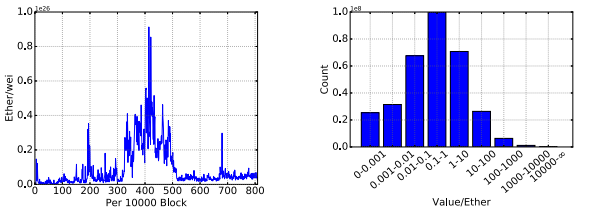
(c) Macro view of GasPrice (d) Micro view of GasPrice

FIGURE 4. Visualization of Dataset 1 (better viewed in color).

use the word cloud to analyze the texts of miners. Fig. 4(a) shows the visualization of the texts of the word cloud. The results show that there are texts left by the mining pool, since most miners are in the mining pool, and they have left their names in the blocks to promote their mining capability. As shown in Table 1, the mean of transaction counts per block is 60.68, and the block time is 15.33 seconds. In other words, the average throughput of Ethereum is about 4 transactions per second. Even when most of the network is active, as shown at 4,900,000 blocks in Fig. 4(b), the throughput is about 16.7 transactions per second. This result implies that Ethereum still has a long way to go to support real-time Internet applications. In Ethereum, a miner has a higher priority to package the transactions with higher "`gasPrice`" into the block. The visualization of "`gasPrice`" is shown in Fig. 4(c) and (d). In a macro view, the "`gasPrice`" is gradually decreasing with the development of the Ethereum community, except for several peaks caused by the extremely frequent transaction when the network is congested. In a micro view, we extract the time from 8,000,000 to 8,020,000 blocks and find that such fluctuations of "`gasPrice`" can be observed by the tidal law. This observation implies that the fluctuations of "`gasPrice`" can potentially be predicted.

TABLE 2. Statistics of Dataset 2

Statistics	Values
No. of Ether Transactions	330,239,865
No. of Addresses	54,688,782
Mean of Amount of Ethers	22.26
Maximum of Amount of Ether	11,901,464.24

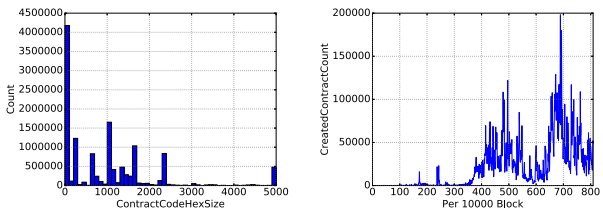


(a) Ether Transferred Amount (b) Ether Transaction Distribution

FIGURE 5. Visualization of Dataset 2.

TABLE 3. Statistics of Dataset 3

Statistics	Values
No. of Created Contracts	16,577,477
No. of Creator Addresses	133,039
No. of Deleted Contracts	5,704,054
No. of Refunded Addresses	19,133,738
Mean of Contract Hex Code Size	962.00



(a) Contract size distribution (b) Count of created contracts

FIGURE 6. Visualization of Dataset 3.

B. DATASET 2: INTERNAL ETHER TRANSACTION

Ether is the native cryptocurrency of Ethereum. The transactions of Ether not only happen in the transactions recorded in the block, but also occur during the smart contract execution. For example, if someone asks a smart contract to send 10 Ethers to another one, the Ether transaction from the contract will not be observed in the block. In some blockchain explorers such as Etherscan,[5] this kind of transactions is also called "*Internal Transaction*". To investigate all the Ether transactions, we process the block and trace data to conduct the internal Ether transaction dataset. As shown in Table 2, 330,239,865 Ether transactions which occur among 54,688,782 addresses are collected. The values of Ether have a large variance, as the maximum is 11,901,464.24 Ethers (about 2 billions dollars now) but the mean is only 22.26 Ethers. Fig. 5(a) presents statistics on the total transaction amount of every 10,000 blocks. It is shown that the most active time for Ether transaction is the time during 4,000,000 to 4,300,000 blocks, matching with the most active time of Initial Coin Offering (ICO). Regarding the Ether distribution as shown in Fig. 5(b), we find that most of Ether transactions fall in the range from 0.1 Ether to 1 Ether, indicating that most of transactions only transfer small amounts of Ethers.

C. DATASET 3: CONTRACT INFO

Ethereum can be considered as a platform for smart contracts. To investigate all the smart contracts on Ethereum, we process the trace data to get the basic information of smart contracts, including the creator, created-time, initial value, contract code, creation code. Some smart contracts can be deleted and refund Ethers to someone if they set a "SUICIDE" operation code inside a function. Therefore, we can observe the actions of contract deletions. According to the statistics in Table 3, there are 16,557,477 smart contracts created by 133,039 addresses. It implies that there should be a number of users who create multiple contracts. An abnormal phenomenon observed from Table 3 is that 5,704,054 contracts are deleted while they refund the Ether balance to 19,133,738 addresses. Generally, a smart contract will not refund Ethers to multiple addresses during deletion. The reason behind this abnormal phenomenon is that Ethereum has suffered from a Denial of Service (DoS) attacks, in which attackers use a vulnerability of the price of "SUICIDE" to create accounts in Ethereum. Before the vulnerability is fixed, a great number of contracts are deleted to direct to empty address, leading to many Ethereum peers shutting down as indicated in previous work [11].

Regarding the contract code, we translate the bytecode into hexadecimal code. Fig. 6(a) gives the statistics of contract size. Particularly, the mean of contract size is 962.00, indicating that the smart contracts take up little space of storage. The contract size distribution also implies that the sizes of most contracts have focused on some clusters. This indicates that many smart contracts may look similar. This similarity will be further investigate in Dataset 4. Fig. 6(b) presents the count of created contracts. It is shown in Fig. 6(b) that the number of new smart contracts is increasing, especially at the time after the concept of "ICO" [12] comes out.

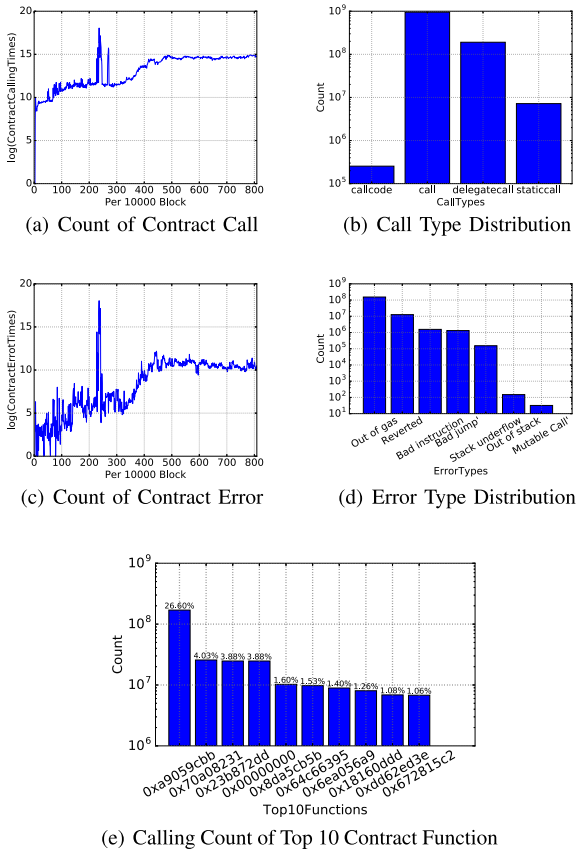
D. DATASET 4: CONTRACT CALL

In EVM, a smart contract can call another one to invoke some codes or functions. To investigate the calls among the Ethereum contracts (which are represented as addresses), we extract Contract Calls in the execution from the trace dataset. The contract call dataset includes the caller, called address, calling function. As shown in Table 4, it consists of 1,148,572,009 Contract Calls, among which 639,336,722 contain input codes and 169,463,261 contain

[5]http://etherscan.io

TABLE 4. Statistics of Dataset 4

Statistics	Values
No. of Contract Calls	1,148,572,009
No. of Calls with Inputs	639,336,722
No. of Calls with Errors	169,463,261



(a) Count of Contract Call

(b) Call Type Distribution

(c) Count of Contract Error

(d) Error Type Distribution

(e) Calling Count of Top 10 Contract Function

FIGURE 7. Visualization of Dataset 4.

errors. Fig. 7 gives the visualization of Contract Calls. In particular, Fig. 7(a) and 7(c) show that, during the time from 2,300,000 to 2,460,000 blocks, contract calls and errors occur very frequently. This is caused by the DoS attacks mentioned in the above subsection, as the attackers invoked a large number of contracts in batches and some of them throw errors. Fig. 7(b) gives the distribution of call types. In particular, Fig. 7(b) shows that most of developers prefer to use "`call`" and "`delegatecall`" rather than "`staticcall`" and "`callcode`," since the logic of "`call`" and "`delegatecall`" is clearer and more practical than other two calls. Fig. 7(d) shows the error types during calling contract, indicating that most of errors are caused by "Out of gas," which is mainly resulted from the wrong settings of message senders. The second most common error is "`Reverted`," which is a manually-thrown exception by the developers. Moreover, other errors such as "Bad instruction" and "Bad jump destination" are often caused by the contract codes themselves. Generally, the compiler

TABLE 5. Statistics of Dataset 5

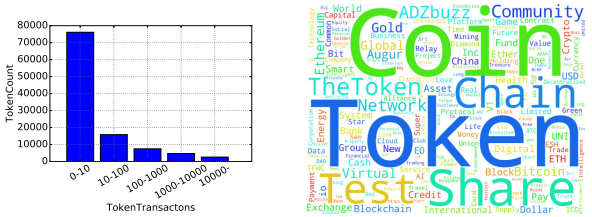
Statistics	Values
No. of ERC20 Contracts	106,683
No. of ERC20 Transactions	227,698,645
No. of Holder Addresses	42,146,575

of smart contracts will use the hash value of function name and parameters as the entry of the function. In other words, in Ethereum smart contracts, the identical function in source code will have the identical entry in the complied contract code. We then count the calling contract functions to see what functions are the most common ones. The distribution of top-10 functions is shown in Fig. 7(e). The results show that most of the calling functions concentrated on some types of them. For example, top-10 functions have occupied 46.32% of the contract calls. Moreover, after verifying the hash values of functions with the open-source contracts, we obtain the functions in source code. We then have the top-3 functions: "`transfer(address,uint256)`,""`balanceOf(address)`" and "`transferFrom(address, address, uint256)`". This result implies that the most common contract calls are about tokens and there might be a great similarity among the contracts due to the similar calls.

E. DATASET 5: ERC20 TOKEN TRANSACTION

From the above analysis, we observe that the most active smart contracts on Ethereum now are the token contracts. We next further investigate the token contracts. In order to collect the information of tokens, we process the receipt dataset to extract the standard events, which are defined in the standard ERC20 protocol of the Ethereum community.[6] Additionally, each ERC20 token contains basic information like name, symbol, total supply, etc. We then send calls to the local Ethereum peers to collect such basic information of ERC20 tokens. As shown in Table 5, 106,683 smart contracts are considered as ERC20 contracts, since they output the events that are defined as the standard ERC20 token transactions. There are 227,698,645 ERC20 transactions among 42,146,575 holder addresses. Generally, the number of holder addresses could be much larger than that of exact human holders because a user may own several addresses. Meanwhile, some token issuers will send the tokens to other users without their permissions (also called *token air-drop* [13]). Fig. 8(a) shows the transaction count distribution for each ERC20 token. We can easily observe the Matthew effect [14] from Fig. 8(a) as most of the token transactions happen in a few token contracts. Fig. 8(b) presents the word cloud of names of ERC20 tokens. It is shown in Fig. 8(b) that the most common words are "Chain," "Coin," and "Share," on which the most ERC20 tokens focus. In addition, another common word is "Test," implying that many ERC20 contracts deployed on Ethereum are just for the testing purpose.

[6]https://eips.ethereum.org/EIPS/eip-20

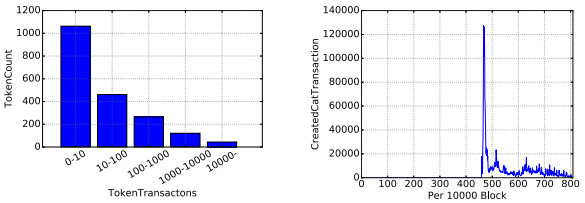


(a) ERC20 Popularity Distribution (b) Word Cloud of ERC20 Tokens

FIGURE 8. Visualization of Dataset 5 (better viewed in color).

TABLE 6. Statistics of Dataset 6

Statistics	Values
No. of ERC721 Contracts	1,954
No. of ERC721 Transactions	7,524,827
No. of Holder Addresses	414,829



(a) ERC721 Popularity Distribution (b) CryptoKitties Turnover Times

FIGURE 9. Visualization of Dataset 6.

F. DATASET 6: ERC721 TOKEN TRANSACTION

ERC721 token is another contract protocol proposed by Ethereum community.[7] Different from ERC20 token, ERC721 token is indivisible. In the contract function, the parameter is not the value of token but the token ID. For example, a virtual pet in smart contract could be a ERC721 token, which is not separable but can be transferred. Table 6 presents the statistics of ERC721 contracts. We find that 1,954 ERC721 contracts contain 7,524,827 token transactions and 414,829 holder addresses. It is worth mentioning that some of the collected contracts do not follow the standard ERC721 protocol exactly. These contracts are also included in the dataset since they output the token transferred events in the receipt. Fig. 9(a) shows the popularity distribution of ERC721 tokens. Compared with ERC20 tokens, the amount of ERC721 tokens is much lower. The major reason is that ERC721 applications require much more workloads on visualization at each token, consequently improving the development difficulty. We also investigate a popular ERC721 token contract called CryptoKitties. It is one of the most famous ERC721 token contracts, selling the virtual cats as tokens. Each cat is represented as a token in the ERC721 contract. We count the turnover times distributed by birth block of the cats, as shown in Fig. 9(b). Fig. 9(b) also shows that the cats that were born in 4,500,000 to 5,000,000 blocks have the higher turnover times than others. At that time, the type of CryptoKitties reaches the peak. The time to obtain the peak in Fig. 9(b) is almost the same as that in Fig. 4(b) and (c), implying that the popularity of CryptoKitties leads to the congestion of Ethereum.

V. APPLICATIONS OF XBLOCK-ETH

This section presents applications of the XBlock-ETH framework. As shown in Fig. 1, the architecture of Ethereum consists of peers, blockchain, smart contracts, and tokens. Thus, we also categorize the applications according to the top 3 layers in this architecture. Meanwhile, we also discuss the research opportunities in each layer.

A. BLOCKCHAIN SYSTEM ANALYSIS

Since XBlock-ETH processes data from realistic blockchain systems, it can be used to support the following applications.

1) DECENTRALIZATION ANALYSIS

Decentralization is one of the key features of blockchain systems. However, there are few studies on the decentralization evaluation of the blockchain systems. In particular, the work of [15] presents the measurement of the mining pool for Bitcoin. Although Gencer *et al.* [16] present a measurement study on the decentralization level of Bitcoin and Ethereum, their study only considers several metrics such as network bandwidth, mining power, and fairness. In contrast, our XBlock-ETH data offers a more comprehensive measurement of Ethereum. Moreover, our work can be used to analyze the decentralization of users, contract owners, and miners. In addition, our XBlock-ETH can also be used to make comparisons with other blockchain systems, such as Bitcoin, EOS, or other blockchain systems.

2) GASPRICE PREDICTION

Since the transaction fees are equal to "`gasPrice`" times "`gasUsed`," the users can control the "`gasUsed`" in a reasonably low range to minimize the transaction fees charged by miners. Meanwhile, we can learn from Section IV-A that there is always a gap between the minimum "`gasPrice`" and the average "`gasPrice`" in a block, leading to the opportunity to save fees. Recent studies such as Other-tech [17], Majuri [18] analyze the "`gasPrice`" of Ethereum while several Ethereum websites (e.g., Etherscan,[8] Etherchain[9]) provide tools to predict the "`gasPrice`" in a short time. However, those tools are essentially *black boxes*, and the accuracy and correctness of them cannot be assured. In summary, the prediction of "`gasPrice`" has great economic value such that the user of Ethereum can save the money or shorten waiting time through the "`gasPrice`" prediction while it is worthwhile to conduct an in-depth study in the future.

[7] https://eips.ethereum.org/EIPS/eip-721

[8] http://etherscan.io

[9] http://etherchain.org

3) PERFORMANCE BENCHMARK

Performance is crucial to blockchain systems. There are a number of studies on blockchain performance optimizations, such as Omniledger [19], Algorand [20], and RapidChain [21]. Meanwhile, some optimized blockchain systems (e.g., Monoxide [22]) adopt the realistic blockchain transaction data to conduct the performance evaluation for blockchain systems. To compare the performance of different optimization methods, a common benchmark of real-world use cases for blockchain systems is needed. Zheng *et al.* [23] and BlockBench [24] propose performance evaluation tools for blockchain systems. The performance benchmark requires simulating the user behaviors and obtaining data similar to real-world blockchain systems. In this context, the XBlock-ETH framework can be regarded as a benchmark since the source data is generated exactly by real-world users.

B. SMART CONTRACT ANALYSIS

As one of the most popular smart contract platforms, Ethereum has attracted a large number of software developers as well as a huge number of smart contracts. Therefore, Ethereum has a more active developer community compared with other smart contract platforms such as EOS and Tron, which claim to have higher throughput and lower latency than Ethereum. Consequently, our XBlock-ETH framework (on top of Ethereum) can be used in the studies of smart contracts. We summarize the potential applications of XBlock-ETH as follows.

1) CONTRACT SIMILARITY AND RECOMMENDATION

As indicated in Section IV, there is a great similarity between the smart contract codes and the call of smart contracts. Code similarity evaluation is a traditional research topic in software engineering as a number of studies concentrate on code similarity detection [25], [26]. Several recent studies focus on the similarity analysis of smart contracts. In particular, Etherscan[8] provides the query system based on similar contracts. Finding similar contracts is beneficial to developers while developing new contracts. For example, developers can estimate user behaviors before publishing the contract. Meanwhile, Huang *et al.* [27] propose the method to recommend differentiated codes to update smart contracts based on the existing codes of smart contracts. In addition, in the aspect of users, recommending similar smart contracts will help users to find the contracts suitable for themselves.

2) CONTRACT DEVELOPER ANALYSIS

Developer analysis that is another traditional research topic in software engineering includes developer network analysis [28], behavior analysis [29], fault prediction [30], and so on. With respect to developer analysis, XBlock-ETH also includes a large network of smart contract developers. For example, there some on-chain libraries deployed and provided by different developers; these libraries can be invoked by others. Each developer can be identified by his/her own Ethereum address. Thus, the contract calling network can also be regarded as the collaboration network of contract developers. The network and structure of developer collaboration may inform us about the reliability of the contract codes. For example, the developer who develops a smart contract with vulnerabilities will have a higher risk of developing new contracts with vulnerabilities than others. In this sense, our XBlock-ETH can be beneficial to the developer analysis after analyzing the smart contracts of developers.

3) CONTRACT VULNERABILITY DETECTION

Blockchain security and privacy have drawn extensive attention recently [31]–[34]. The security of smart contracts has been a hot research topic in the blockchain research community. In particular, the vulnerability of smart contracts has attracted extra attention. A number of malicious attacks on Ethereum (e.g., TheDAO attack) have already resulted in a huge loss (in terms of millions of dollars) [35]. To prevent smart contracts from malicious attacks, the vulnerability detection on contracts is a critical step. There are some recent attempts in vulnerability detection. For example, Oyente [7], Zeus [36], teEther [37], S-gram [38], and ContractFuzzer [39] propose the tools of vulnerability detection on smart contracts. In some cases, the vulnerability detection methods of smart contracts can be inspired and motivated by traditional software vulnerability detection methods as they are essentially equivalent to the verification of the codes. In this aspect, several studies focus on verifying contract codes on blockchains; these contract codes are also called "bytecode" or "opcode". Our XBlock-ETH that essentially includes the data of contract codes can be applied to contract vulnerability detection.

4) FRAUD DETECTION

Due to the huge economic value and the popularity of smart contracts, smart contracts can be exploited by malicious users as scams. For example, crowd-funding contracts with a promised huge return to attract victims for investment. It is reported in [40] that Ponzi scam contracts can defraud others' cryptocurrencies. Several approaches [40]–[43] have been proposed to detect the fraud contracts on Ethereum. Most of the methods are mainly based on the codes and transaction records of smart contracts while they are included in XBlock-ETH data. Thus, XBlock-ETH data can be further leveraged in fraud detection.

C. CRYPTOCURRENCY ANALYSIS

Blockchain-based cryptocurrency has become a hot topic in recent years due to decentralization and the reduced cost. There are a large number of cryptocurrencies in Ethereum, including the Ether, ERC20 tokens, and ERC721 tokens. It is shown in the CoinMarketCap[10] that more than 2,000 kinds of tokens can be used in third-party exchange. Therefore, cryptocurrency analysis based on blockchain data can bring

[10]https://coinmarketcap.com/all/views/all/

huge financial values. We roughly categorize the cryptocurrency analysis into cryptocurrency transferring analysis, cryptocurrency price analysis, and fake user detection, which are explained as follows.

1) CRYPTOCURRENCY TRANSFERRING ANALYSIS

Analysis of cryptocurrency transactions is a preliminary step to conduct cryptocurrency transferring analysis. Regarding Ether transferring, Chen *et al.* [6] propose the graph analysis on Ether transactions and derive some insights from graph analysis. With regard to ERC20/ERC721 tokens, Victor *et al.* [44] and Somin *et al.* [45] propose the analysis of the token trading network. After the analysis of cryptocurrency transactions, further analysis of user behaviors can be done. For example, the users of tokens may form different communities. The community discovery can be conducted through analyzing cryptocurrency transactions. Moreover, the anonymity of blockchain-based cryptocurrency can result in money-laundering behaviors, which can be essentially identified and detected via cryptocurrency transaction analysis. Our XBlock-ETH data offers potential solutions to these issues.

2) CRYPTOCURRENCY PRICE ANALYSIS

The price of blockchain-based cryptocurrencies has been affected by multiple different factors, such as government policies, technology innovations, social sentiment, and business activities. Several recent studies focus on the price analysis and prediction of cryptocurrencies [46]–[48]. The typical cryptocurrency price analysis consists of three steps: (i) collect price data form the cryptocurrency exchanges, (ii) identify the relevance between cryptocurrency prices and other factors, and (iii) forecast the future prices and predict the potential profits. However, the price of cryptocurrencies can sometimes be maliciously controlled by some parties. Thus, the data cleaning process is necessary to obtain accurate and normal cryptocurrency price data. Some of the cryptocurrency price data is stored in the decentralized exchange contracts, which can be used for cryptocurrency price analysis, while the raw data may require further preprocess to benefit future analysis.

3) FAKE USER DETECTION

Fake user detection [49]–[51] is a traditional research topic in social networks. The cryptocurrency users in blockchain systems also form social-network like communities, in which there are also some fake users controlled by the developers to improve the DApps activity rankings. Because the DApp (or cryptocurrency) ranking is based on some metrics related to the user activities, such as Daily Active Users (DAU). Therefore, many developers exploit the loophole to fabricate some fake users to improve activities so as to gain higher rankings. Although some DApp websites, such as DAppReview[11] mark the cryptocurrencies with fake users, this kind of fake user detection is almost done in a black box or manually. In addition, there are few studies on fake user detection on cryptocurrency. The permission-less blockchain systems which are often free of charge may advocate more frequent fake user activities than permissioned blockchain systems. Our XBlock-ETH will be further improved to support fake user detection in the future.

VI. RELATED WORK AND DISCUSSION

Some previous studies on Ethereum data will be described and discussed in this section. We categorize the state-of-the-art literature into two types: *Data tools* and *Data analysis*.

Regarding Ethereum data tools, some studies provide open-source tools or APIs with users to obtain the data. For example, EtherQL [52] offers a query layer for Ethereum. Blocksci [53] constructs a platform for researchers to analyze the blockchain data. DataEther [54] is a tool to obtain the data from Ethereum, with code modification of the Ethereum clients. Google BigQuery[12] imports the data of Bitcoin and Ethereum and enables researchers to analyze the data online while updating Ethereum data has been stopped for a long time. Meanwhile, it is pretty challenging for researchers to download, update, and analyze the blockchain data. There are also some websites offering data APIs for developers to use or analyze, including Amberdata.[13] However, these third-party APIs always restrict the usage rating so that it is difficult for researchers to crawl all the data. In summary, most of these studies only offer tools or APIs to researchers while failing to offer well-processed up-to-date datasets.

Some recent studies provide an analysis of the Ethereum data. For example, studies of [40]–[42] propose the contract classification methods to detect Ponzi schemes. Moreover, Chen *et al.* [6] analyze the transactions and construct three graphs to observe the behaviors on Ethereum. Furthermore, the work of [55] analyzes the ERC20 tokens on Etherem and find the un-standard token. Another popular research area on Ethereum data is smart contract security. For example, Oyente [7], and Zeus [36] propose the security analysis tools for Ethereum smart contracts to find vulnerable codes. Although some of these studies release some datasets, most of them are only suitable for specific research questions. Furthermore, most of them are difficult to be updated. It is worth mentioning that XBlock-ETH does not contain the off-chain data such as the price data in exchanges, the source code of verified smart contracts, the behavior on Github of the DApps even if they are also crucial for the analysis. Since those data are not generated by the Ethereum, we only concentrate on the on-chain data in this paper.

VII. CONCLUSION

This paper introduces a well-processed up-to-date on-chain dataset of Ethereum, namely XBlock-ETH, which includes the data of the Ethereum blockchain, smart contracts, and cryptocurrencies. Moreover, comprehensive statistics and exploration of the datasets are presented. The XBlock-ETH datasets have been released on the XBlock.pro website. Furthermore, the research opportunities of the XBlock-ETH

[11] http://dapp.review

[12] https://cloud.google.com/bigquery/

[13] http://amberdata.io

datasets are also outlined. Our XBlock-ETH is promising to promote the studies on Ethereum. The future improvements are listed as following: (1) More features: The exploration of the basic features of the datasets is given in this paper. Ethereum is a complex ecosystem that includes decentralized finance, stable coin, and so on. More features of the Ethereum data will be explored in the future. (2) Extra off-chain data from Ethereum: The off-chain data is also important since it provides information about the off-chain behaviors of both developers and users. In the future, the off-chain data will be collected. (3) Joint data analysis with other blockchain systems: There are some other blockchain systems that have also attracted a large number of users and developers. The joint data analysis of Ethereum and other permissionless blockchains will be conducted in the future.

REFERENCES

[1] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: A survey," *Int. J. Web Grid Services*, vol. 1, no. 4, pp. 352–375, 2018. [Online]. Available: https://doi.org/10.1504/IJWGS.2018.10016848

[2] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[3] V. Buterin *et al.*, "Ethereum white paper," 2013. [Online]. Available: https://ethereum.org/whitepaper/

[4] V. Buterin and F. Vogelsteller, "Erc20 token standard," 2015. [Online]. Available: https://theethereum.wiki/w/index.php/ERC20_Token_Standard

[5] H.-N. Dai, Z. Zheng, and Y. Zhang, "Blockchain for Internet of Things: A survey," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8076–8094, Oct. 2019.

[6] T. Chen *et al.*, "Understanding ethereum via graph analysis," in *Proc. IEEE Conf. Comput. Commun.*, 2018, pp. 1484–1492.

[7] L. Luu, D.-H. Chu, H. Olickel, P. Saxena, and A. Hobor, "Making smart contracts smarter," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 254–269.

[8] N. Szabo, "The idea of smart contracts," Nick Szabos Papers and Concise Tutorials, vol. 6, 1997.

[9] O. Kharif, "Cryptokitties mania overwhelms ethereum network's processing," *Bloomberg*, 2017. [Online]. Available: https://www.bloomberg.com/news/articles/2017-12-04/cryptokitties-quickly-becomes-most-widely-used-ethereum-app

[10] L. Luu and Y. Velner, "Kybernetwork: A trustless decentralized exchange and payment service," 2017. [Online]. Available: https://whitepaper.io/document/43/kyber-network-whitepaper

[11] T. Chen *et al.*, "An adaptive gas cost mechanism for ethereum to defend against under-priced dos attacks," in *Proc. Int. Conf. Inf. Secur. Practice Experience*, 2017, pp. 3–24.

[12] S. T. Howell, M. Niessner, and D. Yermack, "Initial coin offerings: Financing growth with cryptocurrency token sales," National Bureau of Economic Research, Tech. Rep., 2018. [Online]. Available: https://ecgi.global/working-paper/initial-coin-offerings-financing-growth-cryptocurrency-token-sales-0

[13] P. van Valkenburgh, "A token airdrop may not spare you from securities regulation," 2017. [Online]. Available: https://coincenter.org/link/a-token-airdrop-may-not-spare-you-from-securities-regulation.

[14] R. K. Merton, "The matthew effect in science: The reward and communication systems of science are considered," *Science*, vol. 159, no. 3810, pp. 56–63, 1968.

[15] C. Wang, X. Chu, and Q. Yang, "Measurement and analysis of the bitcoin networks: A view from mining pools," 2019, *arXiv:1902.07549*. [Online]. Available: https://arxiv.org/abs/1902.07549

[16] A. E. Gencer, S. Basu, I. Eyal, R. Van Renesse, and E. G. Sirer, "Decentralization in bitcoin and ethereum networks," 2018, *arXiv:1801.03998*. [Online]. Available: https://arxiv.org/abs/1801.03998

[17] Jin. S, "Ethereum gas price analysis," 2018. [Online]. Available: https://medium.com/onther-tech/ethereum-gas-price-analysis-b70080e2e0d7

[18] Y. Majuri, "Simply explained: Ethereum gas," 2018. [Online]. Available: https://medium.com/@yakko.majuri/blockchain-definition-of-the-week-ethereum-gas-2f976af774ed

[19] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "Omniledger: A secure, scale-out, decentralized ledger via sharding," in *Proc. IEEE Symp. Secur. Privacy*, 2018, pp. 583–598.

[20] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *Proc. 26th Symp. Operating Syst. Principles*, 2017, pp. 51–68.

[21] M. Zamani, M. Movahedi, and M. Raykova, "Rapidchain: Scaling blockchain via full sharding," in *Proc. SIGSAC Conf. Comput. Commun. Secur.*, 2018, pp. 931–948.

[22] J. Wang and H. Wang, "Monoxide: Scale out blockchains with asynchronous consensus zones," in *Proc. 16th USENIX Symp. Networked Syst. Des. Implementation*, 2019, pp. 95–112.

[23] P. Zheng, Z. Zheng, X. Luo, X. Chen, and X. Liu, "A detailed and real-time performance monitoring framework for blockchain systems," in *Proc. 40th Int. Conf. Softw. Eng.: Softw. Eng. Practice*, 2018, pp. 134–143.

[24] T. T. A. Dinh, J. Wang, G. Chen, R. Liu, B. C. Ooi, and K.-L. Tan, "Blockbench: A framework for analyzing private blockchains," in *Proc. ACM Int. Conf. Manage. Data*, 2017, pp. 1085–1100.

[25] M. Chilowicz, E. Duris, and G. Roussel, "Syntax tree fingerprinting for source code similarity detection," in *Proc. IEEE 17th Int. Conf. Program Comprehension*, 2009, pp. 243–247.

[26] L. Luo, J. Ming, D. Wu, P. Liu, and S. Zhu, "Semantics-based obfuscation-resilient binary code similarity comparison with applications to software plagiarism detection," in *Proc. 22nd ACM SIGSOFT Int. Symp. Found. Softw. Eng.*, 2014, pp. 389–400.

[27] Y. Huang, Q. Kong, N. Jia, X. Chen, and Z. Zheng, "Recommending differentiated code to support smart contract update," in *Proc. IEEE 27th Int. Conf. Program Comprehension*, 2019, pp. 260–270.

[28] A. Meneely, L. Williams, W. Snipes, and J. Osborne, "Predicting failures with developer networks and social network analysis," in *Proc. 16th ACM SIGSOFT Int. Symp. Found. Softw. Eng.*, 2008, pp. 13–23.

[29] L. Layman, L. Williams, and R. S. Amant, "Toward reducing fault fix time: Understanding developer behavior for the design of automated fault detection tools," in *Proc. IEEE 1st Int. Symp. Empirical Softw. Eng. Meas.*, 2007, pp. 176–185.

[30] E. J. Weyuker, T. J. Ostrand, and R. M. Bell, "Using developer information as a factor for fault prediction," in *Proc. 3rd Int. Workshop Predictor Models Softw. Eng.*, 2007, pp. 8.

[31] Q. Xu, Z. Su, M. Dai, and S. Yu, "APIS: Privacy-preserving incentive for sensing task allocation in cloud and edge-cooperation mobile Internet of Things with SDN," *IEEE Internet Things J.*, early access, Nov. 19, 2019, doi: 10.1109/JIOT.2019.2954380.

[32] Q. Xu, Z. Su, and R. Lu, "Game theory and reinforcement learning based secure edge caching in mobile social networks," *IEEE Trans. Inf. Forensics Secur.*, early access, Mar. 16, 2020, doi: 10.1109/TIFS.2020.2980823.

[33] Z. Su, Y. Wang, Q. Xu, and N. Zhang, "LVBS: Lightweight vehicular blockchain for secure data sharing in disaster rescue," *IEEE Trans. Dependable Secure Comput.*, early access, Mar. 13, 2020, doi: 10.1109/TDSC.2020.2980255.

[34] Q. Xu, Z. Su, and Q. Yang, "Blockchain-based trustworthy edge caching scheme for mobile cyber-physical system," *IEEE Internet Things J.*, vol. 7, no. 2, pp. 1098–1110, Feb. 2020.

[35] M. I. Mehar *et al.*, "Understanding a revolutionary and flawed grand experiment in blockchain: The dao attack," *J. Cases Inf. Technol.*, vol. 21, no. 1, pp. 19–32, 2019.

[36] S. Kalra, S. Goel, M. Dhawan, and S. Sharma, "Zeus: Analyzing safety of smart contracts," in *Proc. Nat. Down Syndrome Soc.*, 2018, pp. 1–12.

[37] J. Krupp and C. Rossow, "teether: Gnawing at ethereum to automatically exploit smart contracts," in *27th USENIX Secur. Symp., Secur.*, 2018, pp. 1317–1333.

[38] H. Liu, C. Liu, W. Zhao, Y. Jiang, and J. Sun, "S-gram: towards semantic-aware security auditing for ethereum smart contracts," in *Proc. 33rd ACM/IEEE Int. Conf. Automated Softw. Eng.*, 2018, pp. 814–819.

[39] B. Jiang, Y. Liu, and W. Chan, "Contractfuzzer: Fuzzing smart contracts for vulnerability detection," in *Proc. 33rd ACM/IEEE Int. Conf. Automated Softw. Eng.*, 2018, pp. 259–269.

[40] W. Chen, Z. Zheng, J. Cui, E. Ngai, P. Zheng, and Y. Zhou, "Detecting ponzi schemes on ethereum: Towards healthier blockchain technology," in *Proc. 27th Int. Conf. World Wide Web*, 2018, pp. 1409–1418.
[41] M. Bartoletti, S. Carta, T. Cimoli, and R. Saia, "Dissecting ponzi schemes on ethereum: Identification, analysis, and impact," *Future Gener. Comput. Syst.*, vol. 102, pp. 259–277, 2020.
[42] W. Chen, Z. Zheng, E. C.-H. Ngai, P. Zheng, and Y. Zhou, "Exploiting blockchain data to detect smart ponzi schemes on ethereum," *IEEE Access*, vol. 7, pp. 37 575–37 586, 2019.
[43] C. F. Torres, M. Steichen, and R. State, "The art of the scam: Demystifying honeypots in ethereum smart contracts," in *Proc. 28th USENIX Conf. Secur. Symp.*, 2019, pp. 1591–1607. [Online]. Available: http://dl.acm.org/citation.cfm?id=3361338.3361449
[44] F. Victor and B. K. Lüders, "Measuring ethereum-based erc20 token networks," in *Proc. Int. Conf. Financial Cryptography Data Secur.*, 2019, pp. 113–129.
[45] S. Somin, G. Gordon, and Y. Altshuler, "Network analysis of erc20 tokens trading on ethereum blockchain," in *Proc. Int. Conf. Complex Syst.*, 2018, pp. 439–450.
[46] C. Lamon, E. Nielsen, and E. Redondo, "Cryptocurrency price prediction using news and social media sentiment," *SMU Data Sci. Rev*, vol. 1, no. 3, pp. 1–22, 2017.
[47] J. Abraham, D. Higdon, J. Nelson, and J. Ibarra, "Cryptocurrency price prediction using tweet volumes and sentiment analysis," *SMU Data Sci. Rev.*, vol. 1, no. 3, p. 1, 2018.
[48] W. Mensi, K. H. Al-Yahyaee, and S. H. Kang, "Structural breaks and double long memory of cryptocurrency prices: A comparative analysis from bitcoin and ethereum," *Finance Res. Lett.*, vol. 29, pp. 222–230, 2019.
[49] Q. Cao, M. Sirivianos, X. Yang, and T. Pregueiro, "Aiding the detection of fake accounts in large scale social online services," in *Proc. 9th USENIX Conf. Networked Syst. Des. Implementation*, 2012, pp. 15–15.
[50] O. Varol, E. Ferrara, C. A. Davis, F. Menczer, and A. Flammini, "Online human-bot interactions: Detection, estimation, and characterization," in *Proc. 11th Int. AAAI Conf. Web Soc. Media*, Montréal, Québec, Canada, May 15–18, 2017, pp. 280–289. [Online]. Available: https://aaai.org/ocs/index.php/ICWSM/ICWSM17/paper/view/15587
[51] E. Ferrara, O. Varol, C. Davis, F. Menczer, and A. Flammini, "The rise of social bots," *Commun. ACM*, vol. 59, no. 7, pp. 96–104, 2016.
[52] Y. Li, K. Zheng, Y. Yan, Q. Liu, and X. Zhou, "Etherql: A query layer for blockchain system," in *Proc. Int. Conf. Database Syst. Adv. Appl.*, 2017, pp. 556–567.
[53] H. Kalodner, S. Goldfeder, A. Chator, M. Möser, and A. Narayanan, "Blocksci: Design and applications of a blockchain analysis platform," 2017, *arXiv:1709.02489*. [Online]. Available: https://arxiv.org/abs/1709.02489
[54] T. Chen *et al.*, "Dataether: Data exploration framework for ethereum," in *Proc. 39th IEEE Int. Conf. Distrib. Comput. Syst.*, Dallas, TX, USA, Jul. 7–10, 2019, pp. 1369–1380. [Online]. Available: https://doi.org/10.1109/ICDCS.2019.00137
[55] T. Chen *et al.*, "Tokenscope: Automatically detecting inconsistent behaviors of cryptocurrency tokens in ethereum," in *Proc. 26th ACM Conf. Comput. Commun. Secur.*, 2019, pp. 1503–1520.

PEILIN ZHENG is a student at Sun Yat-sen University, Guangzhou, China. His research interests include blockchain monitoring, blockchain optimization, and blockchain-based decentralized applications.

ZIBIN ZHENG is a Professor at Sun Yat-sen University, Guangzhou, China. He received the Ph.D. degree from The Chinese University of Hong Kong in 2011. His research interests include services computing, software engineering, and blockchain. He received the ACM SIGSOFT Distinguished Paper Award at ICSE' 10, Best Student Paper Award at ICWS' 10, and IBM Ph.D. Fellowship Award.

JIAJING WU received the B.Eng. degree in communication engineering from Beijing Jiaotong University, Beijing, China, in 2010 and the Ph.D. degree from Hong Kong Polytechnic University, Hong Kong, in 2014. In 2015, she joined the School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China, where she is currently an Associate Professor. Her current research interest includes blockchain intelligence, network science and cyber-physical systems. She was the recipient of the Hong Kong Ph.D. Fellowship Scheme during her Ph.D. study in Hong Kong from 2010 to 2014. She is serving as an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II: EXPRESS BRIEFS.

HONG-NING DAI is an Associate Professor in Faculty of Information Technology at the Macau University of Science and Technology. He received the Ph.D. degree in computer science and engineering from the Department of Computer Science and Engineering at the Chinese University of Hong Kong. His research interests include wireless networks, mobile computing, and distributed systems. He is serving as an Associate Editor for IEEE ACCESS and *Connection Science* and an Editor of *Ad Hoc Networks* (Elsevier).