IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 14, 2019, accepted March 6, 2019, date of publication March 18, 2019, date of current version April 5, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2905769

Exploiting Blockchain Data to Detect Smart Ponzi Schemes on Ethereum

WEILI CHEN[1,2], ZIBIN ZHENG[1,2], EDITH NGAI[3], PEILIN ZHENG[1,2], AND YUREN ZHOU[1,2]

[1]School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510006, China
[2]National Engineering Research Center of Digital Life, Sun Yat-sen University, Guangzhou 510006, China
[3]Department of Information Technology, Uppsala University, 751 05 Uppsala, Sweden

Corresponding author: Zibin Zheng (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 Grant U1811462 and Grant 61773410, in part by the Guangdong Province Universities and Colleges Pearl River Scholar Funded Scheme (2016), in part by the Program for Guangdong Introducing Innovative and Entrepreneurial Teams under Grant 2016ZT06D211, and in part by the STINT Initiation Grant for International Collaboration under Grant IB2017-6978.

ABSTRACT Blockchain technology becomes increasingly popular. It also attracts scams, for example, a Ponzi scheme, a classic fraud, has been found making a notable amount of money on Blockchain, which has a very negative impact. To help to deal with this issue and to provide reusable research data sets for future research, this paper collects real-world samples and proposes an approach to detect Ponzi schemes implemented as smart contracts (i.e., smart Ponzi schemes) on the blockchain. First, 200 smart Ponzi schemes are obtained by manually checking more than 3,000 open source smart contracts on the Ethereum platform. Then, two kinds of features are extracted from the transaction history and operation codes of the smart contracts. Finally, a classification model is presented to detect smart Ponzi schemes. The extensive experiments show that the proposed model performs better than many traditional classification models and can achieve high accuracy for practical use. By using the proposed approach, we estimate that there are more than 500 smart Ponzi schemes running on Ethereum. Based on these results, we propose to build a uniform platform to evaluate and monitor every created smart contract for early warning of scams.

INDEX TERMS Blockchain, smart contract, Ponzi Schemes, ethereum, data mining.

I. INTRODUCTION

Blockchain technology is described as a disruptive technology that is going to revolutionize many industries. The ongoing discussion of blockchain technology has triggered the attention of policymakers, regulators and the industrial and academic communities [1], [2]. *Blockchain* takes its origins from the famous Bitcoin which makes value transfer between anonymous participants possible without relying on authoritative third-parties by combining many mature technologies such as digital signature schemes, the proof-of-work mechanism, distributed technologies, and so on [3]. Technically, it is a continuously growing list of records of value transfer transactions maintained by a peer-to-peer network through a distributed consensus mechanism. It has many compelling features and is usually referred to as the next generation of Internet [4], as they create an Internet of Value compared with the traditional Internet of Information.

Nowadays, many applications based on blockchain are proposed, such as banking industry [5], Internet of Things (IoT) [6], and smart grids [6]. Many projects aiming to support blockchain-based applications have been created. Ethereum is a famous open-source blockchain based distributed platform. It provides a Turing-complete virtual machine (i.e., Ethereum virtual Machine, EVM) to implement *smart contrasts*. A smart contract, coined by Nick Szabo, is a set of promises and protocols specified in digital form [7], [8]. Smart contracts are easy to implement based on Ethereum platform because of its decentralized distributed characteristics and supporting of many high-level languages, such Solidity.[1] Smart contracts deployed on Ethereum platform can never be tampered with and will automatically enforce when preset conditions are met. Thus, it can be applied in various domains [9]–[11]. Blockchain platforms that support smart contracts are considered as the second-generation blockchain [4].

The associate editor coordinating the review of this manuscript and approving it for publication was Quan Zou.

[1]http://solidity.readthedocs.io/en/develop

New technologies are vulnerable to exploitation by scams. For example, the rise of email attracts a lot of spams. Blockchain, as an emerging technology, also attracts many scams because of its lack of regulation and anonymous characteristic. Many types of scams can be found in the blockchain area, such as exploits, hacks, and phishing [12]. A recent study estimates that more than 7 million USD has been gathered during 9/2/2013 to 9/9/2014 by scams in Bitcoin [13]. Ponzi scheme, a classic fraud named after a notorious fraudster of almost 100 years ago, also has its blockchain-based form [13], [14]. A Ponzi scheme is a fraudulent investment operation where the operator generates returns for older investors through revenue paid by new investors, rather than from legitimate business activities or profits of financial trading [15]. In a Ponzi scheme, many participants, especially those posteriors, are doomed to lose most of their invested money. It has been reported that all kinds of Ponzi schemes are making big money from people who want to participate in the blockchain technology but do not understand how it works [16]–[18]. Obviously, Ponzi schemes hurt the economy and are prohibited in many countries.

Nowadays, many Ponzi schemes disguised themselves under the veil of *smart contracts* [14]. We call these Ponzi schemes as *smart Ponzi schemes* and the corresponding smart contract as *Ponzi scheme contract*. Ponzi scheme contracts have many advantages for operators, such as 1) no maintenance fee is needed after the smart contract implemented; 2) participants have great confidence in continuously paying back as the smart contract cannot be terminated and is automatically enforced; and 3) the operators stay anonymous. Figure 1 displays the propaganda picture of a typical smart Ponzi scheme. The propaganda words are as the following:

> *"Hello! My name is Rubixi! I'm new & verified pyramid smart contract on the Ethereum blockchain. When you send me 1 ether, I will multiply the amount and send it back to your address when the balance is sufficient. My multiplier factor is dynamic (**min. x1.2 max. x3**), thus my payouts are accelerated and **guaranteed** for months to come"*

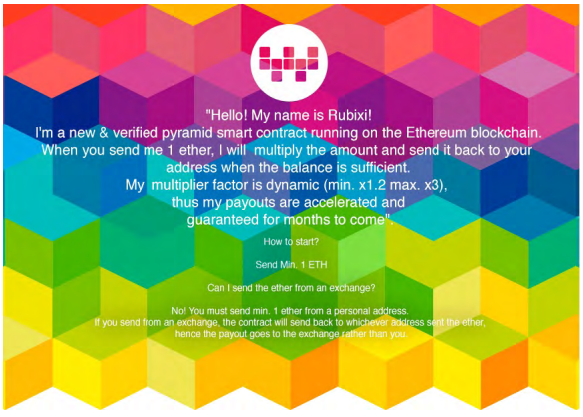

FIGURE 1. The Propaganda Picture of a Smart Ponzi Scheme Source: https://bitcoindtalk.org/index.php?topic=1400536.0.

As publicized, the smart contract was relatively profitable and the returns seem guaranteed to come soon. However, the true situation is far from described. Through manually checking the contract's transaction history, we found that only 19.6% (22 out of 112) participants made a profit from the contract and more than 40% of the profit was taken by two participants. One of the two profit makers was the contract creator. Obviously, this contract causes loss in most of the participants.

The above examples show that detecting blockchain based Ponzi schemes is a significant and urgent task for the following reasons: 1) It is a big challenge for investors and users lacking professional knowledge to understand what blockchain is and detect the scams; and 2) it is imperative for national authorities and regulators to strengthen supervision and legislation for the health and rightfulness of blockchain related markets [19].

To help dealing with this issue, this paper focuses on detecting smart Ponzi schemes implemented on Ethereum. Because blockchain data are publicly accessible, it is possible to address this problem by exploiting the blockchain data. However, detecting smart Ponzi schemes is not an easy job for three reasons: 1) enough verified samples must be collected, and it can only be verified by manually check the source codes, as they disclose the logic of smart contracts; 2) with the development of Ethereum, many smart contracts are implemented on Ethereum each day, which makes it inefficient to manually check each contract; and 3) there are more than two million smart contracts running on Ethereum at the moment, but only around 1% have source codes.[2] (We call all smart contracts having source code as open source contracts, the others as hidden source contracts.) Thus, detecting smart Ponzi schemes through manual checking is neither unrealistic nor comprehensive. Therefore, this paper proposes to establish a classification model not relying on the source code to automatically identify smart Ponzi schemes.

To build an effective model for detecting smart Ponzi schemes without source code, we need 1) enough smart contracts that have *labels* (verified whether they are smart Ponzi schemes) and 2) effective features which can be extracted without source codes. To this end, as shown in Fig. 2, we first obtain all the source codes of open source smart contracts implemented before 9/7/2017 from *Etherescan.io* and manually check whether it is Ponzi scheme contract (i.e., collecting ground truth data). Then, two kinds of features, namely the account features and the code features, of these contracts are extracted from the external transactions, internal transactions and operation codes. Next, a better classification model, as compared with many other methods, is proposed. Finally, the proposed model is applied to detect all the latent smart Ponzi schemes (i.e., smart Ponzi schemes hidden source code).

[2]https://etherscan.io/accounts/c

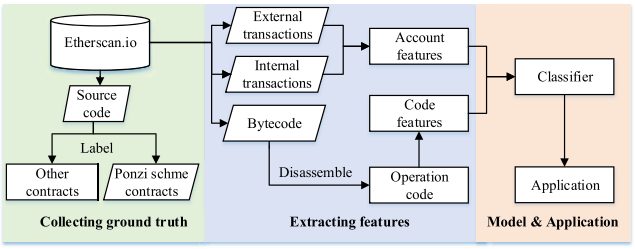

FIGURE 2. The framework of smart ponzi schemes detection.

Extended from its previous conference version [20], which reports preliminary method and results on this issue, the journal version makes new contributions, including: 1) providing more samples by manually checking more open source smart contracts (manually checked 3,780 open source contracts (1,382 in [20]) and collect 200 smart Ponzi schemes (131 in [20]); 2) a better model is proposed and the efficiency of the proposed model is demonstrated by extensive experimental results; 3) six new account features are extracted for better classification; 4) the problem of sample imbalance is considered in this new version and the proposed model is compared with a commonly used framework for this problem; 5) this paper adds to the body of literature in blockchain fraud detection by comparing the smart Ponzi scheme detection problem with the anomaly detection problem and evaluating the efficiency of the proposed model in comparison to many methods. Furthermore, the new model is used to estimate the number of smart Ponzi schemes created each month.

The remaining of this paper is organized as follows. Section II provides a brief introduction of the Ethereum platform, some key concepts and the related work. A detailed description of the data, the extracted features, and the classification model are presented in Section III. Experimental results, analysis, and application are summarized in Section IV. Finally, we conclude the paper in Section V.

II. ETHEREUM, SMART CONTRACTS AND RELATED WORK

This section briefly introduces Ethereum and smart contracts. First of all, we introduce the Ethereum platform and its state transition mechanism. Then, a source code snippet of a Ponzi scheme contract is provided and analyzed. Finally, the operation code, the mnemonic form of bytecode, and the main source of features in our model, is introduced.

A. ETHEREUM IN A NUTSHELL

Ethereum is a famous blockchain platform [8]. Unlike Bitcoin, it provides an Ethereum Virtual Machine (EVM) that can execute code of arbitrary algorithmic complexity. The EVM is the runtime environment for EVM *bytecode*, which is a Turing-complete programming language. To help build applications, many new high-level languages such as *Solidity*[3] are presented. It takes only a few lines of code to create a simple distributed application with Ethereum platform.

[3] http://solidity.readthedocs.io/en/develop

Smart contracts, a famous idea coined by Nick Szabo in 1996, aim to provide new ways to formalize business or personal relationships [7]. It becomes very convenient to implement smart contracts based on the Ethereum platform because of EVM. Technically, it takes only three steps to implement a smart contract based on Ethereum: 1) write the source code of the smart contract with high-level languages, such as *Solidty*; 2) compile the source code into bytecode using an EVM compiler; and 3) upload the bytecode to the Ethereum blockchain with an Ethereum client.

From a technical point of view, a blockchain system can be considered as a state transition and maintenance system [8]. In Ethereum blockchain, the state comprises all the *accounts* and their status. An account, or an *address*, is a string of digits and characters that can be shared with anyone who wants to interact with the account. Two types of accounts can be found in Ethereum: *external accounts* that are controlled by users (i.e., a human) and *contract accounts* that are controlled by the bytecode stored together with the account. Regardless of the types of the accounts, they are treated equally by the EVM. Every account has two fields which determine a unique status of the account: *nonce* (indicates the number of transactions sent from that address) and *balance*.

A *transaction* is usually a message sending from one account to another with binary data (its *payload*) or Ether (the crypto-fuel for the Ethereum network). Transferring Ether and calling a function in a contract are two typical transactions initiated by users. Besides, new transactions may be triggered by existing transactions. For example, a victim who invests some Ether to a Ponzi scheme contract account may trigger Ether transfer transactions from the contract account to previous victims. We call the transactions of sending Ether to a contract account as *external transactions* (or investments) and the triggered transfer transactions as *internal transactions* (or payments) of the contract.

An external transaction can be seen as a trigger for the operation of smart contracts. It is usually initiated by a user (i.e., external accounts) with some binary data to invoke a function in the contract. The internal transactions is a kind of response to an external transaction. For example, suppose a contract B has a function which will transfer some Ether to accounts C and D when the contract receives some Ether from other accounts. Then, if account A sending 5 Ether to account B, it will trigger two transfer transactions from B to C and D. In this example, the transaction from A to B is a external transaction of contract B; and the transactions from B to C and D are two internal transactions of contract B. Thus, external and internal transactions can be seen as invoke-responses for a smart contract, which may be useful in reflecting the logic of the contract. Actually, we extracted account features from the transactions for smart Ponzi scheme detection.

Transactions occurred in a certain period of time, which have been validated by *miners*, or the maintenance nodes of the Ethereum network, are packaged into blocks and appended into a public and append-only ledger, i.e., the blockchain. Each transaction is charged with a certain amount

of *gas*, according to the consumed resources used for validating the transaction, to encourage miners to maintain the blockchain and to prevent system abuse. The gas multiplied by the user-defined *Gasprice* is the *fee* for the transaction, which has to be paid upfront from the sending account and depleted gradually according to specific rules. The leftover gas will be refunded in the same way after the execution. If the gas is used up before the completion of the execution, the transaction is failed and all the side effects are reverted. Thus, it is very important for the creator to provide enough gas to ensure the success of a transaction.

As each transaction is executed and packaged independently by *miners*, a consensus protocol must be employed to maintain a unique state of Ethereum blockchain in a certain time period. Consensus protocol is the foundation of mutual trust between users in the blockchain system. Ethereum adopts a similar consensus protocol as Bitcoin. This ensures that the Ethereum system will be safe unless a particular attacker owning 51% computing power of the whole network. Based on the security of the system, trust is built among Ethereum users for its transparent code execution process.

B. A SOURCE CODE SNIPPET OF A SMART PONZI SCHEME

In the Ethereum platform, Solidity is a commonly used language for writing smart contracts. It is a contract-oriented, high-level language whose syntax is similar to that of JavaScript. In this subsection, we present a simplified Ponzi scheme contract written with Solidity (Fig. 3) to show how to identify a Ponzi scheme contract by exploiting its source code. This is also the way we classify the downloaded open source smart contracts and collect positive samples.

The code snippet shown in Fig. 3 is extracted from the contract named *Rubixi* for understanding why it is a smart Ponzi scheme. The code snippet begins with the keyword *contract* to define a smart contract named Rubixi. Generally speaking, the body of a smart contract consists of two parts: variables and functions. Functions can be called by transactions or messages from other accounts. During the execution of a function, the data stored in that contract can be renewed.

The codes from line 2 to 12 are variable definition used to record the key information of the contract. For example, *balance* records the current balance of the smart contract and the *creator* records the address of the creator. The struct *Participant* records an investor's address and its corresponding reward. The dynamically-sized array of ''Participant'' structs are used to store all the investors of the contract. Other variables are contract-specific and will be explained in the following.

The function Rubixi defined from line 14 to 16 is the constructor which assigns the *creator* with the address of the contract creator. It runs only once when the contract is created. The function with no name in line 17, which contains only a function call to *addPayout*, is called *fallback*

```
1  contract Rubixi {
2    uint private balance = 0;
3    uint private collectedFees = 0;
4    uint private feePercent = 10;
5    uint private Order = 0;
6    uint private pyramidMultiplier = 300;
7    address private creator ;
8    struct Participant {
9    address etherAddress ;
10   uint payout ;
11   }
12   Participant [] private participants ;
13
14   function Rubixi ( ) {
15     creator = msg . sender ;
16   }
17   function ( ) { addPayout ();}
18
19   function addPayout ( ) private {
20     uint fee = feePercent ;
21     participants.push(Participant(msg.sender ,
22     (msg.value * pyramidMultiplier ) / 100));
23     if ( participants.length == 10)
24     pyramidMultiplier = 200;
25     else if (participants.length == 25)
26     pyramidMultiplier = 150;
27     balance += (msg.value * (100 - fee ) ) / 100;
28     collectedFees += (msg.value * fee ) / 100;
29     while ( balance > participants[Order].payout) {
30       uint payoutToSend = participants[Order].payout ;
31       participants[Order].etherAddress.send(payoutToSend);
32       balance -= participants[Order].payout ;
33       Order += 1;
34     }
35   }
36
37   function collectAllFees ( ) onlyowner {
38   if ( collectedFees == 0) throw ;
39   creator.send ( collectedFees ) ;
40   collectedFees = 0;
41   }
42 }
```

FIGURE 3. A simplified ponzi scheme contract(Rubixi).

function. It is executed when receiving Ether sending from other accounts. Thus, an investment transaction will trigger the function *addPayout*, defined in line 19 to 35. This function is the key of the contract as it implements the main logic of a Ponzi scheme. Firstly, it records the address and the reward to payout of the investor in order (line 21–22). Then, it calculates fees (line 28). Finally, it pays to previous investors while the balance is enough (line 29–34).

The array *participants* defined in line 12 records all the investors in order, including address (*msg.sender*) and reward to payout (*payout=msg.value *pyramidMultiplier)/100*). Note that the propagated high profit (see Fig. 1) is controlled by the variable *pyramidMultiplier*, which was first set to 300 (line 6), but then reduced to 200 (line 23) from the 10th participant and 150 (line 25) from the 25th participant. Obviously, to attract early participants, the contract owner promised a higher profit for them. It is worth mentioning that *pyramidMultiplier* should be set to above 100 to be profitable. The while loop from line 29 to line 34 tries to pay all the previous participants by their investment order until the balance is not enough.

Taking fees from participants is the main purpose of operating a Ponzi scheme. Figure 3 shows that the creator of Rubixi charges every investment 10 percent, and the fees are

collected by calling the function *collectAllFees* in line 37. Note that the function can only be called by the owner of the contract (i.e., the creator).

Figure 3 clearly shows the logic of the contract, which obviously can be identified as a Ponzi scheme. As seen from the code snippet, two kinds of transactions may occur with a smart contract: external transactions and internal transactions. The essence of a smart Ponzi scheme determines the payment order of the internal transactions. Thus, it is possible to identify a smart Ponzi scheme from its transactions history.

C. DEPLOY A CONTRACT

As mentioned in subsection II-A, an Ethereum contract is a series of ''Ethereum virtual machine code'' or ''EVM code'' residing in the Ethereum blockchain. We call this the *bytecode* of a contract. To deploy a smart contract, one should first write the source code of the smart contract with a high-level language (e.g., Solidity), then compile the source code into EVM bytecode. It consists of a series of bytes. Each byte is an operation (or the operand of operation). Each operation has a corresponding mnemonic form for human readability. For example, the mnemonic form of EVM bytecode 0×10 is LT, which means less-than comparison. We call LT and such mnemonic form of EVM bytecode as *opcode*. The appendix of Ethereum yellow paper [21] contains a complete list of the EVM bytecode and its mnemonic form, i.e., opcode. A disassembler[4] can be used to get the operation code of a contract from bytecode.

To make the contract callable from other accounts, the bytecode should be deployed in the main Ethereum network. A special transaction targeted to the zero-account (the account with address 0) creates a new contract. The bytecode of the contract provided as the payload of that transaction will be executed; the result will be stored with the new contract account and be record permanently in the blockchain. The address of the new contract will return to the creator, which can then be shared with others. Note that only the bytecode is needed to deploy a smart contract, thus the creator can hide the source code of the contract. As a matter of fact, most smart contracts are deployed without source code according to etherscan.io. Thus, it may be impossible for a user to identify latent smart Ponzi schemes (i.e., smart Ponzi schemes hidden source codes).

D. RELATED WORK

Blockchain technology is described as a disruptive technology that is going to revolutionize many industries. Thus, it has become a research hotspot. Three types of research can be found in the literature. The first type focuses on the underlying techniques, such as consensus mechanisms [22]–[24], performance improvement and evaluation [25], [26], and smart contract [27]. The second type discusses the application of blockchain technology, such as finance service [28], [29], Internet of Things (IoT) [10], [30], and smart traveling [31]. More information can be found in the survey [32]. The last but the most related type of work is data mining on blockchain. Thanks to the publicly accessible characteristic, blockchain provides an unprecedented opportunity for data analysis to answer various questions, for example, usage characteristics [33]–[35], anonymity [36], [37] and economic behavior analysis [38], [39].

With the development of the Internet, online ''High-yield'' investment program (HYIP) became a typical form of Ponzi schemes. A preliminary analysis was provided on economic aspects of it by using data collected from HYIP websites [40]. More detailed research can be found in [41], where a model was set out to estimate the turnover and profits of HYIPs. Both papers focused on HYIPs which use centralized virtual currencies. With the development of blockchain technology, various scams in the name of technical innovation upsurge inevitably. As a previous research pointed out [42], many investors are approaching blockchain technology in the hope of getting on a promising investment vehicle. However, it is difficult for a ordinary investor to understand blockchain technology and detect any scams on it. Thus, many scams take advantage of this information asymmetry. Marie Vasek and Tyler Moore present an empirical analysis of Bitcoin-based scams [13]. Four groups of scams including Ponzi schemes, mining scams, scam wallets, and fraudulent exchanges were identified. It is reported that 13,000 distinct victims were found and at least $11 million has been contributed to the 192 scams. Data mining technologies have been used for the detection of financial fraud [43] and it is used for detecting Bitcoin Ponzi schemes [44]. A recent study focused on the economic aspects of smart Ponzi schemes [14]. They use *normalized Levenshtein distance* [45] as a measure of similarity between bytecodes to detect hidden smart Ponzi schemes. Different from their study, this paper focused on identifying smart Ponzi schemes with bytecodes by using machine learning and data mining methods.

III. DATA, FEATURE EXTRACTION AND CLASSIFICATION MODEL

This section provides an overview of the data, the extracted features, and the classification model.

A. DATA

1) GROUND TRUTH

Due to the source code of most smart contracts are hidden, we must build a model without source code. However, we need ground truth data which can be verified by manually checking to train the model. To this end, we collect all the open source code smart contract created before 9/7/2017 from the *etherscan.io* and manually check whether they are Ponzi scheme contracts. In this way, 200 Ponzi scheme contracts and 3580 non-Ponzi scheme contracts are identified. These manually checked smart contracts are then used as ground truth data to build the classification model. In order to make the model applicable to all smart contracts, the features are

[4]https://etherscan.io/opcode-tool

extracted without source codes. For example, features are obtained from the transaction history and the bytecode.

2) TRANSACTION HISTORY

Two kinds of transactions (i.e., external transactions and internal transactions) are collected and used in this paper. Since external transactions are stored in the blockchain, it is easy to collect by running an Ethereum client to synchronize all the data. However, internal transactions result from the execution of smart contracts and do not store in the blockchain. In our previous work [20], we collect these transactions by crawling the Ethereum explorer website *etherscan.io*. However, only the last 10,000 transactions of an account can be download due to the restriction of the website. To address this issue, we modify an Ethereum client and replay all the transactions. By doing this, we collect all the transaction history from the launch of Ethereum on 7/30/2015 to 9/7/2017. Our dataset of Ponzi schemes and all the transaction history are available at https://pan.baidu.com/s/1TNBPjubIDc0GJ1kLKjiJyw. After collecting the data, the external and internal transactions are cleaned and stored into one table. Each row in the table records a transaction. There are many fields providing corresponding information of a transaction. In particular, an *is Error* field indicates whether the transaction is failed. By using this table, we can extract all the transactions referred to a contract. We use all the successful transactions in our model.

3) BYTECODE

Bytecode is the "body" of a smart contract and is stored with the account after deploying by a special transaction in the blockchain. Thus, we can get all the bytecodes of smart contracts from the transaction history. To make the model applicable to all contracts, we extract features from the bytecode (note that we do not need source code in feature extraction). However, bytecode is a string of numbers, which is difficult to extract effective features. Thus, we disassemble the bytecode into operation code and extract features from it. Specifically, bytecodes are results of source code compiling using the Ethereum client and the operation code is disassembled from the bytecode. Figure 4 shows the front segment of the bytecode and the operation code of the Rubixi contract which displayed in Figure 3. As one can see, the operation code is composed of instruction mnemonics (such as PUSH1) and its operands (such as 0x60). Please note that some instructions do not have an operand. The opcodes (i.e., instruction mnemonics) are useful in classifying the function of the contract as they indicate all possible operations of the contract [46], [47]. Thus, it is used as the source of code features in this study. It is worth noting that byte n-grams features can be extracted from bytecode without any reverse engineering, and may be a good choice for smart Ponzi scheme detection [48].

```
6060604052600080      1  PUSH1 0x60
8055600181905560      2  PUSH1 0x40
0a60025561012c60      3  MSTORE
0355600481905561      4  CALLDATASIZE
09db90819061002d      5  ISZERO
90396000f3606060      6  PUSH2 0x00b9
```

FIGURE 4. The front fragment of bytecode (left) and operation code (right) of rubixi.

B. ACCOUNT FEATURES

Due to the fraudulent essence, smart Ponzi schemes have several distinct characteristics compared with other contracts: 1) Ponzi scheme contracts usually send Ether to those invested into them; 2) some accounts receive more counts of payment than its counts of investment, for example, the creator who charges fees frequently from the contract; and 3) to keep an image of fast and high return, smart Ponzi schemes may pay back to the investors once they have enough balance, which may result in a low balance.

The intrinsic characteristics of a smart Ponzi scheme determine its special behavior, which in return can be used to judge whether it is a smart Ponzi scheme. As the main behavior of a contract can be reflected by its transactions with other accounts, we manually investigated the transaction history of two typical smart contracts: Rubixi and LooneyLottery. Rubixi is the introduced smart Ponzi scheme (Fig. 1) while LooneyLottery[5] is a typical lottery game contract. Through inspecting the Rubixi transaction history, we found that there are 112 participants involved in the contract, but most payment transactions pertain to the first 25 participants, which means that the majority of the subsequent participants lost their Ether. Furthermore, only the first two participants make a profit from the contract, one of them is the creator who keeps collecting fees from the contract. When analyzing the transaction history of LooneyLottery, we found distinct random characteristics. Just as its name suggests, we found 22 participants having 733 transactions with the amount larger than zero, but only 13 (0.18%) transactions actually pay to its participants from the contract; besides, the payees appear to be random. This phenomenon shows that users participant in the contract many times in the hope of receiving a random reward.

The analysis of the transaction history of Rubixi gives some initial impression of smart Ponzi schemes, such as, 1) the payment transactions usually occur after investment transactions, which indicates that the contract usually pays to "known" accounts; 2) many participants receive nothing from the contract; 3) some participants have more payment transactions than investment transactions. Based on these observations and characteristics, we extract 13 key features referred as account features. For the convenience of description, we introduce *difference vectors* $v1$ and $v2$ to describe the difference in counts and amounts between payment and investment for all participants in a contract. Specifically, suppose that there are p participants pertaining to the contract,

[5] http://the.looney.farm/game/lottery

$v1$ and $v2$ are vectors with length p. The $i-$th element of $v1$ is $v1[i] = n_i - m_i$, where m_i *and* n_i denote the counts of investments and payments of the $i-$th participant. Similar to $v1$, the $i-$th element of $v2$ is the difference of the amount of investments and payments of the ith participant. The extracted features are as follows:

- *Balance (Bal):* the balance of the smart contract.
- *N_maxpay (N_max):* the max number of payments to all participants.
- *N_investment (N_Inv):* the number of investments to the contract.
- *N_payment (N_pay):* the number of payments from the contract.
- *Paid one (P1):* the proportion of investors who received at least one payment.
- *Known rate (Kr):* the proportion of receivers who have invested before payment. A high *Kr* means the contract interact more with accounts already knew. We expect with very high Kr of smart Ponzi schemes.
- *Difference counts mean (Dcm):* the mean of the difference vector $v1$.
- *Difference counts standard deviation (Dcsd):* the standard deviation of the difference vector $v1$.
- *Difference counts skewness (Dcs):* the skewness of the difference vector $v1$.
- *Difference amounts mean (Dam):* the mean of the difference vector $v2$.
- *Difference amounts standard deviation (Dasd):* the standard deviation of the difference vector $v2$.
- *Difference amounts skewness (Das):* the skewness of the difference vector $v2$.
- *Paid rate (Pr):* the quotient of N_payment divided by N_investment.

The mean, standard deviation, and skewness of the vector $v1$ and $v2$ are the newly introduced features (except for *Dcs*) as compared with the conference version [20]. The main reason for introducing these new features is that the vectors (i.e., $v1$ and $v2$) depict the difference of income and expenditure of the investors, which is the key for judging whether it is a Ponzi scheme. Actually, the feature *Dcs* (denotes as *D_ind*) has been proven effective in [20].

Table 1 shows three statistics of the extracted features: mean, median and standard deviation (Sd). The table contains two parts: the upper part is the result of all Ponzi scheme contracts and the bottom part is non-Ponzi scheme contracts.

As seen clearly from the table, the statistics between Ponzi scheme contracts and non-Ponzi scheme contracts are hugely different. The first thing to observe is that the standard deviations of all features in non-Ponzi scheme contracts are larger than in Ponzi scheme contracts. This indicates smart Ponzi schemes may behave similarly which results in lower standard deviation. Another notable thing is that the median of many features are very small as compared with the mean, indicating that the distribution of the features may be heterogeneous among the accounts.

The huge difference in the statistics of some features implies that smart Ponzi schemes are a special kind of smart contracts. For example, the mean of balance in the ordinary contract is larger than that in Ponzi scheme contract. The number of participants in common contracts (N_max, N_inv, N_pay) is obviously more than Ponzi scheme contracts. The huge difference between the feature *Dam* vividly shows that the two types of contracts have remarkable difference in Ether flow.

C. CODE FEATURES

The operation code is successful in analyzing the latent problem of a smart contract as it reflects the logic of the smart contract from the aspect of Ethereum Virtual Machine (EVM) [46], [47]. We expect that features extracted from operation code are useful in detecting smart Ponzi schemes. To this end, we extract all the opcodes (i.e., instruction mnemonics) and calculated their frequency. Each opcode with its frequency is regarded as a feature. Please note that we ignore the digit after the opcode, for example, PUSH1 and PUSH2 are considered the same and denote as PUSH. Sixty-four different opcodes are found in the 3,780 contracts' operation codes. Thus, the code feature is 64 dimension. We have noticed that some opcodes appear frequently in all the contracts (in Figure 5 of the conference version [20], we showed the opcode clouds of two contracts without three frequent opcodes, i.e., PUSH, DUP and SWAP), but we recommend to include all the opcodes as the source of code features because the overall feature space is only 64 dimension and it may be hard to defined a proper parameter to exclude the frequent opcodes.

TABLE 1. Statistics of extracted account features.

Ponzi Scheme contracts													
	Bal	N_max	N_inv	N_pay	P1	Kr	Dcm	Dcsd	Dcs	Dam	Dasd	Das	Pr
Mean	93.63	81.38	326.24	148.56	0.27	0.53	-100.15	4.44×10^{5}	-0.98	-0.44	1167.69	0.01	0.53
Median	0.00	0.12	63.00	2.00	0.16	0.75	-2.00	28.75	-0.30	0.00	0.00	0.00	0.06
Sd	1193.81	468.74	1366.49	884.57	0.32	0.48	683.02	4.44×10^{6}	6.81	5.49	8177.73	4.46	1.26
Non-Ponzi Scheme Contracts													
	Bal	N_max	N_inv	N_pay	P1	Kr	Dcm	Dcsd	Dcs	Dam	Dasd	Das	Pr
Mean	1214.65	3.45×10^{27}	2621.06	1822.79	0.21	0.21	-71.31	3.28×10^{6}	-0.73	1.51×10^{25}	1.45×10^{9}	1.03	0.60
Median	0.00	0.00	6.00	0.00	0.00	0.00	-1.00	1.00	0.00	0.00	0.00	0.00	0.00
Sd	28944.53	2.07×10^{29}	21199.76	20778.51	0.32	0.37	1547.70	1.05×10^{8}	14.52	9.03×10^{26}	8.02×10^{10}	11.59	2.46

FIGURE 5. The opcode cloud in ponzi scheme (left) and Non-ponzi scheme contract (right).

Figure 5 shows the word cloud graph of opcodes Ponzi contracts and non-Ponzi contracts. Each word is an opcode with its font size representing the used frequency. As one can see, the frequency of opcodes is heterogeneous. The mostly used four opcodes in the two categories are PUSH, DUP, SWAP, and JUMP. These opcodes are used more than 200 times on average in the contracts, the main reason is that the EVM is not a register machine but a stack machine, so all computations are performed in a data area called the stack and these opcodes are all stack-related. For example, PUSH is used for placing items and DUP for duplicate items in the stack. The least used opcode in Ponzi scheme contracts is CREATE, which is used only 20 times in the 200 Ponzi scheme contracts. It makes sense because CREATE is used to create a new account with associated code, and there is almost no reason for a Ponzi scheme contract to create a new account. The least used opcode in non-Ponzi scheme contract is SMOD, the signed modulo remainder operation, which was used in only 222 contracts in the 3580 non-Ponzi contract.

D. CLASSIFICATION MODEL

In the conference version of this paper [20], we use a boosting-based algorithm (i.e., XGBoost [49]) as the classifier (please note that there are other boosting-based methods, for example, AdaBoost [50]), however, three characteristics of the data set prompt us to find a more robust model. First, in this new version, we expanded the sample set with more newly created smart contracts (1,382→3,780), which makes the sample set more diverse. Second, unlike traditional classification problems, detecting Ponzi scheme contract encounters the problem of sample imbalance, as only a small fraction of smart contracts are Ponzi schemes. Finally, informed by [14], there are four types of Ponzi scheme contracts (i.e., array-based pyramid schemes, tree-based pyramid schemes, handover schemes, and waterfall schemes). Thus, in this journal version, we adopt a bagging-based algorithm (i.e., Random Forest [51]), which is proved to be more efficiency in reducing variance as compared with boosting based algorithms [52] (please note that there are other bagging-based algorithms like UnderBagging and SMOTEBagging methods [53]).

A Random Forest (RF) [51] is a combination of Decision Trees, trained by the training sets obtained by the bagging method. In this study, we use pasting method [54], in which the training sets are sampled without replacement from the dataset). Suppose there are N smart contracts in the dataset $\{(x_i, y_i) \mid i = 1, 2, \ldots, N\}$, where $x_i \in R^d$ is the extracted features associated with the $i-$th smart contract, $y_i \in \{0, 1\}$ is the classification label, such that $y_i = 1$ if and only if the smart contract is a verified Ponzi scheme contract. We adopt the following steps to detect a Ponzi scheme contract:

1) Produce N training sets which are independent and identically distributed;
2) N decision trees are trained independently by each training sets. Traditional decision tree selects the best decision attribute from all the attributes of the current node to grow the tree while Random Forest randomly select m attributes from all the attributes of the current node and then selects the best attribute from the m attributes;
3) The final result is obtained through voting by the N Decision Trees.

IV. EXPERIMENTAL RESULTS AND FEATURE ANALYSIS

In this section, we present our experimental results. First, we describe the experiment settings and evaluation metrics. Then, the experiments of the proposed method and other competing methods based on the comparison of the two categories of extracted features are summarized. Finally, the most discriminative features are analyzed and the mode is used to estimate the number of smart Ponzi schemes created before 9/7/2017.

A. EXPERIMENT SETTING

1) DATASETS

Two datasets used in this study. The first dataset consists of all the labels and features of the 3780 open source contracts. This dataset is used to build a classification model. To compare the discriminative power of the two categories of features, three kinds of features: account features, code features, and their combination are used in the experiments. Another dataset contains code features of all the smart contracts (including open source contracts and hidden source contracts) which is used to approximate the number of smart Ponzi schemes running on Ethereum platform. To build the model, we split the corresponding dataset into 80% for training and 20% for testing. The experiments conducted for ten times and the average results are summarized in Table 2.

2) BASELINES

The method is compared with a variety of competing methods grouped into the following categories:

Classification methods. To detect Ponzi scheme contract from others can be seen as a typical two-class classification problem, thus we compared the method with two traditional classification methods: decision trees (DT) [55] and support vector machine (SVM) [56]. Besides, XGBoost [49], which is used in the conference version of the paper [20], is also compared with our method.

TABLE 2. A performannce comparison.

Algorithm	Precision			Recall			F-score		
	Account	Code	All	Account	Code	All	Account	Code	All
IF	0.04	0.03	0.02	0.09	0.06	0.05	0.06	0.04	0.04
OCSVM	0.07	0.06	0.05	0.76	1.00	1.00	0.13	0.10	0.10
OCSVM+DT	0.58	0.65	0.33	0.64	0.72	0.21	0.61	0.68	0.25
OCSVM+SVM	0.41	0.95	0.91	0.04	0.41	0.16	0.07	0.57	0.27
SVM	0.32	0.95	0.91	0.06	0.43	0.16	0.09	0.59	0.27
DT	0.58	0.64	0.31	0.64	0.73	0.24	0.60	0.68	0.27
XGBoost	0.59	0.91	0.90	0.22	0.73	0.67	0.32	0.81	0.76
RF	**0.64**	**0.94**	**0.95**	**0.20**	**0.73**	**0.69**	**0.30**	**0.82**	**0.79**

Anomaly detection. Ponzi scheme contracts might be seen as anomaly due to they account for only a small fraction of all the smart contracts. Thus, anomaly detection methods may be used to identify these contracts. Two commonly used methods: One-class SVM (OCSVM) [57] and Isolation Forest (IF) [58] are used.

Two-step methods. Two-step methods are proved useful in dealing with the unbalance of the data [59]. It adopts unsupervised learning method (for example, One-class SVM in this paper) to alleviate the problem of imbalance and then further use the supervised methods (DT and SVM in this paper) to classify samples as the second step.

3) EVALUATION METRICS

To evaluate the performance of the models, we use the three commonly used metrics as follows:

$$Precision = \frac{true\ positive}{true\ positive + false\ positive}$$

$$Recall = \frac{true\ positive}{true\ positive + false\ negative}$$

$$F - score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

B. RESULTS SUMMARY

Table 2 summarizes the performances of the proposed method and the baselines listed above with the three kinds of features in detecting Smart Ponzi schemes. For the XGBoost model, the combination optimization of three important parameters, which includes learning_rate, max_depth and n_eastimators, is carried out. Other parameters are default values. The combinatorial optimization space is the product of the three parameter spaces, i.e., learning_rate$\bigotimes$max_depth$\bigotimes$n_eastimators, where learning_rate= {0.1, 0.2, 0.3}, max_depth = {3,6,8,9} and n_eastimators = {100, 110, 120, $\cdots$, 200}. The best combination is learning_rate=0.2, max_depth=9 and n_eastimators=180. For the RF (Random Forest), we adopt a similar strategy, the combinatorial optimization space is n_eastimators$\bigotimes$bootstrap$\bigotimes$criterion, where n_eastimators= {20, 30, 40, $\cdots$, 200}, bootstrap={ture, false} and criterion= {gini, entropy}. The best combination is n_eastimators=120, bootstrap=False, criterion=entropy.

Several conclusions can be made from the table. First, the proposed method (i.e., RF) significantly improves the performance of all metrics as compared with XGBoost based on code features. Especially, the precision is raised to 0.95, which indicates RF is an ideal method for smart Ponzi scheme detection. In contrast, all the metrics of the two anomaly detection methods are very low, meaning that Ponzi scheme contracts detection cannot be seen as a traditional anomaly detection problem. As for the two-step methods, OCSVM+SVM seems good in precision, however, the low recall causes a huge reduction in efficiency, because it will miss a lot of true Ponzi scheme contracts. It is worth mentioning that the recall of OCSVM seems relatively high, but this is just an illusion because the precision is too low to make a meaningful prediction. Second, the low F-scores of all methods based on the account features indicate that they cannot be used alone in detecting Ponzi scheme contract. One possible reason of this result may be that many smart contracts are experimental, which makes it hard to detect their types from behavior. Another reason, as compared with code features, may be the number of account features is too small. Third, though the account features cannot be used alone, it may help improving the model's precision, and the precision is more important in the smart Ponzi scheme detection problem. Thus, it may be useful in further analysis. Finally, as based on just code features being good enough, the proposed model can be used to detect Ponzi scheme contracts at the moment of its creation.

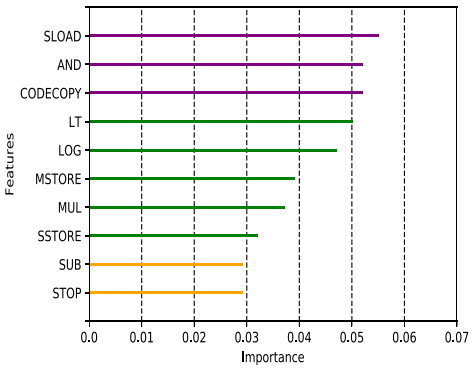

FIGURE 6. The importance of the ten most significant code features of random forest.

C. IMPORTANT FEATURES

As code features are good enough to detect Ponzi scheme contracts, we list the ten most significant code features of random forest in Fig. 6. The description of the opcodes in the graph are summarized in Table 3.

TABLE 3. Opcode and its description.

Opcode	Description
SLOAD	Load word from storage.
AND	Bitwise AND operation.
CODECOPY	Copy code running in current environment to memory.
LT	Less-than comparison.
LOG	Append log record with 0∼4 topics.
MSTORE	Save word to memory.
MUL	Multiplication operation.
SSTORE	Save word to storage.
SUB	Subtraction operation.
STOP	Halts execution.

Fig. 6 shows that the most significant features of random forest and their contributions in detecting Ponzi scheme contract. However, to understand why these features are effective is difficult. Roughly speaking, two classes of opcodes are extracted in these features. One class is storage or memory-related instructions, including SLOAD, CODECOPY, MSTORE, and SSTORE. Another class involves two kinds of operations, i.e., arithmetic operations (including MUL and SUB) and bitwise logic operations (including AND and LT). One possible reason for the effectiveness of the two categories of instructions is that the main function of smart contracts are realized by various operations on some data, thus the numbers of these instructions contained in the smart contracts indicate the type of the contract.

D. ESTIMATION THE NUMBER OF SMART PONZI SCHEMES

One application of our model is to estimate how many smart Ponzi schemes are running on Ethereum. To this end, all the bytecodes of the smart contracts created before 9/7/2017 are extracted from the transaction history. In this way, we obtained 1.58 million contracts and their corresponding bytecodes in total. Then, the bytecodes are disassembled into operation codes and the code features are extracted. Finally, we predict all the smart contracts with our model based on the code features. The result indicates that there are 394 smart Ponzi schemes. Figure 7 shows the number of detected smart Ponzi schemes with corresponding probabilities. As shown from the figure, many detected smart Ponzi schemes have relatively high probability, indicating that the problem of smart Ponzi schemes is more serious than what has been estimated before [14].

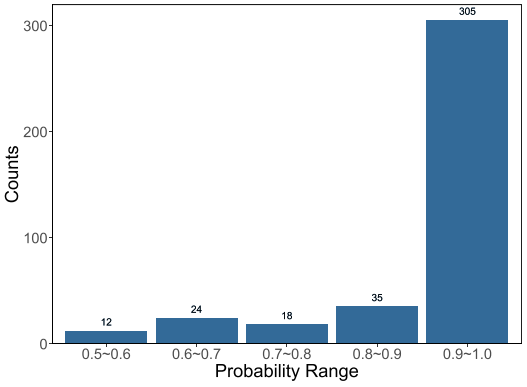

FIGURE 7. The number of detected smart ponzi schemes with corresponding probability range.

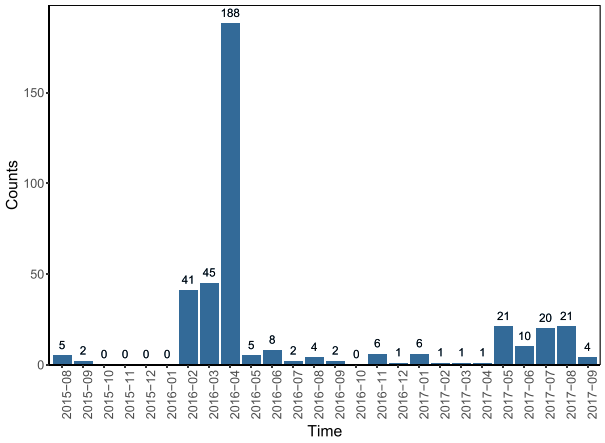

FIGURE 8. The number of detected smart ponzi schemes created each month.

Figure 8 shows the number of detected smart Ponzi schemes created in each month. We find that many smart Ponzi schemes created during February to April 2016. Although there are not many smart Ponzi schemes created in other months, we should note that the number began to increase again after May 2017. This phenomenon indicates that there may be more smart Ponzi schemes nowadays.

Taking the precision and recall of the model into account, it is estimated that there are 507 ($394 \times precision/recall$) smart Ponzi schemes created on Ethereum platform before 9/7/2017, accounting for 0.03% of all the contracts. Although the proportion is low, the number cannot be ignored. We should pay more attention to these contracts, access the impacts and take measures when necessary.

V. CONCLUSION AND FUTURE WORK

Financial scams based on blockchain and cryptocurrency have become an important research problem. With the development of blockchain technology, Ponzi schemes are now under the veil of smart contracts. In this study, we propose a machine learning method to detect smart Ponzi schemes. First of all, the ground truth data is obtained by manually checking the source code. Then, two categories of features, i.e., account features and code features, are extracted. Finally, based on the features and ground truth data, a random forest model is built and applied to identify latent smart Ponzi schemes. We estimate that there are more than 500 smart Ponzi schemes

on Ethereum. It is worth mentioning that the code features are extracted without source code, thus the proposed model can be used to evaluate any contract at the moment of its creation.

In the future, we are going to further this study in three aspects. Firstly, to build more accurate classification model, especially model based on account features and byte n-grams features. Secondly, to study the impacts and types of the detected smart Ponzi schemes. Thirdly, to build a unified platform to evaluate every smart contract for early warning of scams. It is necessary to promote security in the development of blockchain technology.

REFERENCES

[1] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: A survey," *Int. J. Web Grid Serv.*, vol. 14, no. 4, p. 352, 2018.

[2] M. Swan, *Blockchain: Blueprint for a New Economy*. Newton, MA, USA: O'Reilly Media, 2015.

[3] *Bitcoin: A Peer-to-Peer Electronic Cash System*. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[4] CoinDesk. (2017). *Understanding Ethereum-blockchain Research Report*. [Online]. Available: www.coindesk.com/research/understanding-ethereum-report/

[5] Y. Guo and C. Liang, "Blockchain application and outlook in the banking industry," *Financial Innov.*, vol. 2, no. 1, p. 24, 2016.

[6] N. Z. Aitzhan and D. Svetinovic, "Security and privacy in decentralized energy trading through multi-signatures, blockchain and anonymous messaging streams," *IEEE Trans. Depend. Sec. Comput.*, vol. 15, no. 5, pp. 840–852, Sep./Oct. 2018.

[7] N. Szabo. (Sep. 1996). *Smart Contracts: Building Blocks for Digital Markets*. [Online]. Available: http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart_contracts_2.html

[8] *A Next-Generation Smart Contract and Decentralized Application Platform*. [Online]. Available: https://github.com/ethereumlwiki/wiki/White-Paper

[9] A. Norta, "Creation of smart-contracting collaborations for decentralized autonomous organizations," in *Proc. Int. Conf. Bus. Inform. Res.* Cham, Switzerland: Springer, 2015, pp. 3–17.

[10] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the internet of things," *IEEE Access*, vol. 4, pp. 2292–2303, May 2016.

[11] A. Juels, A. Kosba, and E. Shi, "The ring of gyges: Investigating the future of criminal smart contracts," in *Proc. Int. Conf. Comput. Commun. Secur.*, 2016, pp. 283–295.

[12] Chainalysis. (Aug. 2017). *The Rise of Cybercrime on Ethereum*. [Online]. Available: https://blog.chainalysis.com/the-rise-of-cybercrime-on-ethereum/

[13] M. Vasek and T. Moore, "There's no free lunch, even using bitcoin: Tracking the popularity and profits of virtual currency scams," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.* Berlin, Germany: Springer, 2015, pp. 44–61.

[14] M. Bartoletti, S. Carta, T. Cimoli, and R. Saia. (2017). "Dissecting ponzi schemes on ethereum: Identification, analysis, and impact." [Online]. Available: https://arxiv.org/abs/1703.03779

[15] Wikipedia. (Oct. 2017). *Ponzi Scheme*. [Online]. Available: https://en.wikipedia.org/wiki/Ponzi_scheme

[16] S. Higgins. (Oct. 1, 2015). *SEC Seizes Assets from Alleged Altcoin Pyramid Scheme*. [Online]. Available: https://www.coindesk.com/sec-seizes-alleged-altcoin-pyramid-scheme

[17] G. Keirns. (Mar. 15, 2017). *'Gemcoin' Ponzi Scheme Operator Hit with $74 Million Judgment*. [Online]. Available: https://bitcoinwiki.co/gemcoin-ponzi-scheme-operator-hit-with-74-million-judgment/

[18] D. Z. Morris. (May 31, 2017). *The Rise of Cryptocurrency Ponzi Schemes*. [Online]. Available: https://www.theatlantic.com/technology/archive/2017/05/cryptocurrency-ponzi-schemes/528624/

[19] C. K. Elwell, M. M. Murphy, and M. V. Seitzinger. (Dec. 20, 2013). *Bitcoin: Questions, Answers, and Analysis of Legal Issues*. [Online]. Available: https://digital.library.unt.edu/ark:/67531/metadc272070/

[20] W. Chen, Z. Zheng, J. Cui, E. Ngai, P. Zheng, and Y. Zhou, "Detecting ponzi schemes on ethereum: Towards healthier blockchain technology," in *Proc. World Wide Web Conf. World Wide Web*, 2018, pp. 1409–1418.

[21] G. Wood. *Ethereum: A Secure Decentralised Generalised Transaction Ledger*. [Online]. Available: http://gavwood.com/paper.pdf

[22] S. King and S. Nadal. *Ppcoin: Peer-to-Peer Crypto-Currency with Proof–897 of-Stake*. [Online]. Available: https://bitcoin.peryaudo.org/vendor/peercoin-paper.pdf

[23] Hyperledger. (2015). *Hyperledger Project*. [Online]. Available: https://www.hyperledger.org

[24] D. Schwartz, N. Youngs, and A. Britto. *The Ripple Protocol Consensus 901 Algorithm*. [Online]. Available: https://www.cryptiaexchange.com/Whitepaper_Ripple.pdf

[25] W. Chen, Z. Zheng, M. Ma, P. He, P. Zheng, and Y. Zhou, "Poster: Efficient blockchain-based software systems via hierarchical bucket tree," in *Proc. 40th Int. Conf. Softw. Eng., Companion (ICSE-Companion)*, May/Jun. 2018, pp. 360–361.

[26] 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. Pract. Track (ICSE-SEIP)*, May/Jun. 2018, pp. 134–143.

[27] N. Atzei, M. Bartoletti, and T. Cimoli, "A survey of attacks on Ethereum smart contracts (SoK)," in *Principles of Security and Trust*. Berlin, Germany: Springer, 2017, pp. 164–186.

[28] G. W. Peters and E. Panayi, *Understanding Modern Banking Ledgers Through Blockchain Technologies: Future of Transaction Processing and Smart Contracts on the Internet of Money*. Cham, Switzerland: Springer, 2016, pp. 239–278.

[29] D. Kondor, M. Pósfai, I. Csabai, and G. Vattay. (2016). *From 'Blockchain Hype' to a Real Business Case for Financial Markets*. [Online]. Available: https://ssrn.com/abstract=2760184

[30] M. Conoscenti, A. Vetrò, and J. C. D. Martin, "Blockchain for the internet of things: A systematic literature review," in *Proc. IEEE/ACS 13th Int. Conf. Comput. Syst. Appl.*, Dec. 2016, pp. 1–6.

[31] W. Chen, M. Ma, Y. Ye, Z. Zheng, and Y. Zhou, "IoT service based on jointcloud blockchain: The case study of smart traveling," in *Proc. IEEE Symp. Service-Oriented Syst. Eng. (SOSE)*, Mar. 2018, pp. 216–221.

[32] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," in *Proc. Int. Congr. Big Data*, Jun. 2017, pp. 557–564.

[33] S. Meiklejohn *et al.*, "A fistful of bitcoins: Characterizing payments among men with no names," *Commun. ACM*, vol. 59, no. 4, pp. 86–93, 2016.

[34] A. Yelowitz and M. Wilson, "Characteristics of bitcoin users: An analysis of Google search data," *Appl. Econ. Lett.*, vol. 22, no. 13, pp. 1030–1036, 2015.

[35] D. Ron and A. Shamir, "Quantitative analysis of the full bitcoin transaction graph," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.* Berlin, Germany: Springer, 2013, pp. 6–24.

[36] E. Androulaki, G. Karame, M. Roeschlin, T. Scherer, and S. Capkun, "Evaluating user privacy in bitcoin," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.* Berlin, Germany: Springer, 2013, pp. 34–51.

[37] F. Reid and M. Harrigan, "An analysis of anonymity in the bitcoin system," in *Security and privacy in Social Networks*. New York, NY, USA: Springer, 2013, pp. 197–223.

[38] D. Kondor and M. Pósfai, I. Csabai, and G. Vattay, "Do the rich get richer? An empirical analysis of the bitcoin transaction network," *PLoS ONE*, vol. 9, no. 2, 2014, Art. no. e86197.

[39] D. Kondor, I. Csabai, and J. Szüle, M. Pósfai, and G. Vattay, "Inferring the interplay between network structure and market effects in bitcoin," *New J. Phys.*, vol. 16, no. 12, 2014, Art. no. 125003.

[40] T. Moore, J. Han, and R. Clayton, "The postmodern ponzi scheme: Empirical analysis of high-yield investment programs," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.*, vol. 7397. Berlin, Germany: Springer, 2012, pp. 41–56.

[41] J. Neisius and R. Clayton, "Orchestrated crime: The high yield investment fraud ecosystem," in *Proc. APWG Symp. Electron. Crime Res. (eCrime)*. Sep. 2014, pp. 48–58.

[42] F. Glaser, K. Zimmermann, M. Haferkorn, M. C. Weber, and M. Siering, "Bitcoin-asset or currency? Revealing users,hidden intentions," in *Proc. 22th Eur. Conf. Inf. Syst.*, 2014, pp. 1–14.

[43] E. W. T. Ngai, Y. Hu, Y. H. Wong, Y. Chen, and X. Sun, "The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature," *Decision Support Syst.*, vol. 50, no. 3, pp. 559–569, 2011.

[44] M. Bartoletti, B. Pes, and S. Serusi, "Data mining for detecting bitcoin ponzi schemes," in *Proc. Crypto Valley Conf. Blockchain Technol. (CVCBT)*, Jun. 2018, pp. 75–84.

[45] L. Yujian and L. Bo, "A normalized levenshtein distance metric," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 6, pp. 1091–1095, 2007.

[46] N. Atzei, M. Bartoletti, and T. Cimoli, "A survey of attacks on ethereum smart contracts (SoK)," in *Proc. Int. Conf. Princ. Secur. Trust*, Springer, 2017, pp. 164–186.
[47] T. Chen, X. Li, X. Luo, and X. Zhang, "Under-optimized smart contracts devour your money," in *Proc. IEEE 24th Int. Conf. Softw. Anal., Evol. Reeng. (SANER)*, Feb. 2017, pp. 442–446.
[48] E. Raff *et al.*, "An investigation of byte n-gram features for malware classification," *J. Comput. Virology Hacking Techn.*, vol. 14, no. 1, pp. 1–20, 2018.
[49] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 785–794.
[50] D. P. Solomatine and D. L. Shrestha, "AdaBoost.RT: A boosting algorithm for regression problems," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, vol. 2, 2004, pp. 1163–1168.
[51] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.
[52] P. Bühlmann and B. Yu, "Analyzing bagging," *Ann. Statist.*, vol. 30, no. 4, pp. 927–961, 2002.
[53] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera, "A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 4, pp. 463–484, Jul. 2012.
[54] L. Breiman, "Pasting small votes for classification in large databases and on-line," *Mach. Learn.*, vol. 36, nos. 1–2, pp. 85–103, 1999.
[55] J. R. Quinlan, "Induction of decision trees," *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, 1986.
[56] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intell. Syst. Appl.*, vol. 13, no. 4, pp. 18–28, 1998.
[57] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Comput.*, vol. 13, no. 7, pp. 1443–1471, 2001.
[58] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *Proc. 8th IEEE Int. Conf. Data Mining*, 2008, pp. 413–422.
[59] B. Du, C. Liu, W. Zhou, Z. Hou, and H. Xiong, "Catch me if you can: Detecting pickpocket suspects from large-scale transit records," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 87–96.

WEILI CHEN received the M.S. degree in probability and statistics from Central South University, Changsha, China. He is currently pursuing the Ph.D. degree with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China. His research interests include blockchain, data mining, and machine learning.

ZIBIN ZHENG received the Ph.D. degree from the Chinese University of Hong Kong, in 2011. He is currently a Professor of data and computer science with Sun Yat-sen University, China. He serves as the Chair of the Software Engineering Department, Pearl River Young Scholars, and the Founding Chair of the Services Society Young Scientists Forum (SSYSF). In the past five years, he published over 120 international journal and conference papers, including three ESI highly-cited papers, 40 ACM/IEEE TRANSACTIONS papers. According to Google Scholar, his papers have more than 6300 citations, with an H-index of 41. His research interests include blockchain, services computing, software engineering, and financial big data. He was a recipient of several awards, including the outstanding Thesis Award of CUHK, in 2012, the ACM SIGSOFT Distinguished Paper Award at ICSE2010, the Best Student Paper Award at ICWS2010, and IBM Ph.D. Fellowship Award. He served as CollaborateCom'16 General Co-Chair, ICIOT'18 PC Co-Chair, and IoV'14 PC Co-Chair.

EDITH NGAI received the Ph.D. degree from The Chinese University of Hong Kong, in 2007. She held a postdoctoral position with Imperial College London, U.K., from 2007 to 2008. She is currently an Associate Professor with the Department of Information Technology, Uppsala University, Sweden. She is also a Guest Researcher with Ericsson Research, Sweden, from 2015 to 2017. Previously, she has conducted research with Simon Fraser University, Tsinghua University, and UCLA. Her research interests include the Internet of Things, cloud computing, network security and privacy, smart city, and urban computing. She is a Senior Member of ACM. She is a VINNMER Fellow (2009) awarded by the Swedish Governmental Research Funding Agency VINNOVA. Her co-authored papers have received best paper runner-up awards in IEEE IWQoS 2010 and ACM/IEEE IPSN 2013. She was a Program Chair of ACM womENcourage 2015, the TPC Co-Chair of the IEEE SmartCity 2015, the IEEE ISSNIP 2015, and ICNC 2018 Network Algorithm and Performance Evaluation Symposium. She is an Associate Editor for the IEEE ACCESS, the IEEE INTERNET OF THINGS JOURNAL, and the IEEE TRANSACTIONS OF INDUSTRIAL INFORMATICS.

PEILIN ZHENG is currently pursuing the Ph.D. degree with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China. His research interests include performance monitoring and evaluation on blockchain, optimization of smart contracts, and blockchain-based decentralized applications.

YUREN ZHOU received the B. Sc. degree in mathematics from Peking University, Beijing, China, in 1988, and the M.Sc. and Ph.D. degrees in computer science in mathematics from Wuhan University, Wuhan, China, in 1991 and 2003, respectively. He is currently a Professor with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China. His current research interests include design and analysis of algorithms, evolutionary computation, and social networks.