Detecting Ponzi Schemes on Ethereum: Towards Healthier Blockchain Technology

Conference Paper · April 2018

DOI: 10.1145/3178876.3186046

Weili Chen
Sun Yat-Sen University
3 PUBLICATIONS **2** CITATIONS

Detecting Ponzi Schemes on Ethereum: Towards Healthier Blockchain Technology

Weili Chen
School of Data and Computer Science,
Sun Yat-sen University
Guangzhou, China
chenwli9@mail2.sysu.edu.cn

Zibin Zheng
School of Data and Computer Science,
Sun Yat-sen University
Guangzhou, China
Key Laboratory of Machine Intelligence and Advanced Computing (Sun Yat-sen University),
Ministry of Education
zhzibin@mail.sysu.edu.cn

Jiahui Cui
School of Data and Computer Science,
Sun Yat-sen University
Guangzhou, China
cuijh6@mail2.sysu.edu.cn

Edith Ngai
Department of Information Technology, Uppsala University
Uppsala, Sweden
edith.ngai@it.uu.se

Peilin Zheng
School of Data and Computer Science,
Sun Yat-sen University
Guangzhou, China
zhengpl3@mail2.sysu.edu.cn

Yuren Zhou
School of Data and Computer Science,
Sun Yat-sen University
Guangzhou, China
zhouyuren@mail.sysu.edu.cn

ABSTRACT

Blockchain technology becomes increasingly popular. It also attracts scams, for example, Ponzi scheme, a classic fraud, has been found making a notable amount of money on Blockchain, which has a very negative impact. To help dealing with this issue, this paper proposes an approach to detect Ponzi schemes on blockchain by using data mining and machine learning methods. By verifying smart contracts on Ethereum, we first extract features from user accounts and operation codes of the smart contracts and then build a classification model to detect latent Ponzi schemes implemented as smart contracts. The experimental results show that the proposed approach can achieve high accuracy for practical use. More importantly, the approach can be used to detect Ponzi schemes even at the moment of its creation. By using the proposed approach, we estimate that there are more than 400 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.

KEYWORDS

Blockchain; Smart Contract; Ponzi Schemes; Ethereum

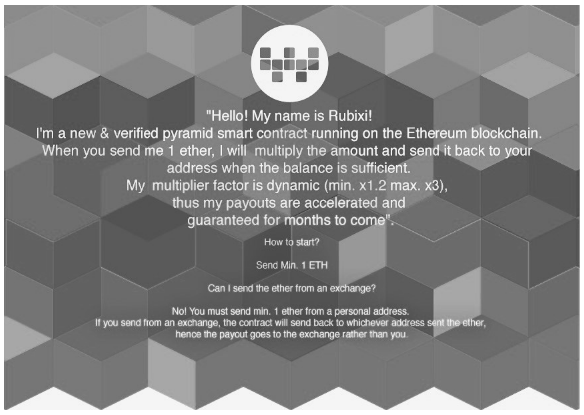
ACM Reference Format:
Weili Chen, Zibin Zheng, Jiahui Cui, Edith Ngai, Peilin Zheng, and Yuren Zhou. 2018. Detecting Ponzi Schemes on Ethereum: Towards Healthier Blockchain Technology. In *WWW 2018: The 2018 Web Conference, April 23–27, 2018, Lyon, France.* ACM, New York, NY, USA, Article 4, 10 pages. https://doi.org/10.1145/3178876.3186046

1 INTRODUCTION

The creation of Bitcoin makes value transfer between anonymous participants possible without relying on authoritative third-parties [27]. Bitcoin combines many mature technologies such as digital signature schemes, the proof-of-work mechanism, distributed technologies, and so on. *Blockchain*, an important part of Bitcoin, is a continuously growing list of records of value transfer transactions maintained by a peer-to-peer network through a distributed consensus mechanism. Blockchain is now a hotspot in both academia and industry [36, 43]. Bitcoin and the derived blockchain technology are usually referred as the next generation of Internet [9], as they create an Internet of Value compared with the traditional Internet of Information.

Many projects based on blockchain have been created. Ethereum is an open-source blockchain based distributed platform. The corresponding coin of Ethereum is called *Ether*. Ethereum provides a Turing-complete virtual machine to execute *smart contrasts*. A smart contract is a computer protocol between mutually distrusting participants, which is automatically enforced on the blockchain when preset conditions are met [4, 37]. The execution of smart contracts cannot be terminated and does not rely on trusted authorities. Smart contracts can be applied in various domains [7, 15, 30]. Blockchain platforms that support smart contracts are considered as the second-generation blockchain [9].

Any new technologies are vulnerable to exploitation by scams. For example, the rise of email creates a lot of spams. Blockchain, as an emerging technology, also attracts many scams because of its lack of regulation and anonymous characteristic. Ponzi scheme, a classic fraud named after a notorious fraudster of almost 100 years ago, also has its blockchain-based form [3, 38]. 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 [39]. In Ponzi scheme, many participants, especially those posteriors, will lose most of their invested money. Obviously, Ponzi schemes hurt the economy and are prohibited in many countries. It has been reported that all kinds of Ponzi schemes are making big money off people who want to participate in the blockchain technology but do not understand how it works [16, 25, 35]. A recent study estimates that more than 7 million USD has been gathered during September 2, 2013 to September 9, 2014 by scams in Bitcoin [38].



"Hello! My name is Rubixi!
I'm a new & verified pyramid smart contract running 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".

How to start?

Send Min. 1 ETH

Can I send the ether from an exchange?

No! You must send min. 1 ether from a personal address.
If you send from an exchange, the contract will send back to whichever address sent the ether, hence the payout goes to the exchange rather than you.

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

Nowadays, many Ponzi schemes disguised themselves under the veil of *smart contracts* [3]. We refer these Ponzi schemes as *smart* Ponzi schemes and the corresponding smart contract *Ponzi scheme contract*. Since participants' confidence in continuously paying back is a key factor in successful operation of Ponzi schemes, smart contract seems to be an attractive tool for Ponzi schemes as it is automatically enforced and cannot be terminated on blockchain. More importantly, the promoters stay anonymous. Fig. 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"*

As publicized, the smart contract was relatively profitable and the returns seem guaranteed to come soon. However, the true payment is far from described. Through manually checking the contract transactions, we found that there are 112 participants in the contracts, but only 22 participants make a profit from the contract. The two luckiest participants take more than 40% of the profit, while the creator is one of them. Obviously, this contract hurts most other participants.

Examples introduced above vividly show that detecting blockchain-based Ponzi schemes is an urgent task. Although some users are perfectly conscious of the possible loss when participating in Ponzi schemes [24, 29], detecting latent Ponzi schemes is still a challenge, because 1) users and investors find it complicated to understand what blockchain is, so they may let alone the fraudulent scams behind it; 2) for national authorities and regulators, the blockchain technology is very new and somehow lives in the gray area of legal system [12]. Therefore, to make the blockchain related markets healthy and rightful, developing technologies to detect blockchain-based Ponzi schemes is not only urgent but also crucial.

Detecting blockchain-based Ponzi schemes is not an easy problem as users of blockchain are essentially anonymous. Given that a smart contract is composed of code, it is possible to manually check whether a smart contract is a Ponzi scheme by going through its source code. However, what makes things worse is that the source codes of smart contracts may be hidden. In fact, only bytecodes are needed for a smart contract to be implemented on the Ethereum blockchain. There are now more than one million smart contracts running on Ethereum, but only less than four thousands of them have source codes[1]. It implies that not only the creator but also the logic of a latent smart Ponzi scheme is hidden. This raises many questions: How many smart Ponzi schemes exist on Ethereum? What types of smart Ponzi schemes are there? What are their characteristics? How much is the influence of smart Ponzi schemes? Before answering these questions, the first and most important question to answer is: how to detect smart Ponzi schemes without source codes?

To establish an effective model for detecting smart Ponzi schemes without source codes, we need 1) enough contracts that have *labels* (determined 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 download 3071 verified smart contracts including normal transactions, fired transactions and source codes. Then the source codes were compiled to bytecodes and the bytecodes dissembled to operation codes (See Section 2 for detailed information). Next, the account features are extracted from the transactions; the code features are extracted from the operation codes. Finally, a classification model based on XGBoost is proposed.

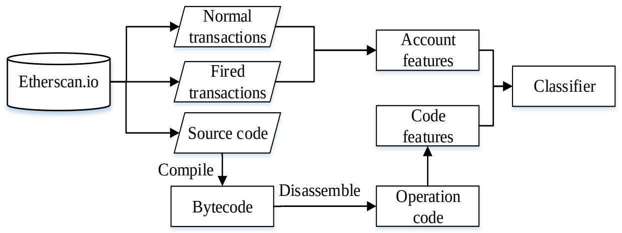


Figure 2: The Framework of Smart Ponzi Schemes Detection

A key contribution of the paper is an experimental validation of the feasibility of detecting smart Ponzi schemes at the moment of its creation. Because we extracted code features from operation codes which is public accessible after the contract deployed. This result is of great significance because by using the model we can: 1) detect smart Ponzi schemes even the source codes are intentional hidden; 2) detect smart Ponzi schemes before it causing any damage; and 3) build a risk warning platform based on it.

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

The remaining of this paper is organized as follows. Section 2 provides a brief introduction of the Etherem platform and some key concepts. A detailed description of the data, the extracted features and the classification model are presented in Section 3. Experimental results and analysis are summarized in Section 4. Finally, we summarize the related work in Section 5 and conclude the paper in Section 6.

2 ETHEREUM AND SMART CONTRACTS

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

2.1 Ethereum in a Nutshell

Ethereum is a blockchain platform with an Ethereum Virtual Machine (EVM), which can execute code of arbitrary algorithmic complexity and allows users to create blockchain applications by a few lines of code [4].

Technically, the EVM is the runtime environment for EVM bytecode, which is a Turing-complete programming language. Smart contracts based on Ethereum is a series of EVM bytecode. It takes three steps to create a smart contract based on Ethereum: 1) write the smart contract source code in a high-level language, for example, *Solidity*[2]; 2) compile the source code into bytecode using the EVM compiler; and 3) upload the bytecode to the blockchain with an Ethereum client.

From the technical standpoint, a blockchain system can be considered as a state transition and maintenance system [4]. In Ethereum system, the state consists of all *accounts* - in twenty-byte address[3]. The account contains four fields to determine a unique status of it. The four fields are the nonce, the Ether balance, the contract code and the storage. The first two fields usually change when a transaction occurs. A transaction is usually a message sending from one account to another with binary data (its *payload*) or Ether. Take account A sending 5 Ether to account B as an example, this is a typical transaction, which will reduce the balance of A and increase the balance of B by 5 Ether. The nonce of each account will increase one to make sure that the transaction can be processed only once. If presented, the code will execute with the payload as input data and the storage as temporary space. For example, a sending Ether transaction from a victim to a Ponzi scheme contract may trigger payment transactions to previous participants. Under this situation, we call the sending Ether transaction as *normal transaction* and the triggered payment transactions as *fired transactions*.

In Ethereum, transactions occurred in a certain period of time are packaged into blocks and appended into a public and append-only ledger, i.e., the blockchain. Each transaction is executed independently by *miners*, the maintenance nodes of the Ethereum network. Miners are mutually untrusted peers and they can validate execution result from other miners. To encourage miners to maintain the blockchain, each transaction is charged with a certain amount of *fees*, according to the consumed resources used for validating the transaction. The consumed *Gas* (a special unit used in Ethereum to measure how much work a transaction takes to perform) multiplying the user-defined *Gasprice* is the fee for the transaction, which is same as the reward for the miners. The execution of a transaction may be failed, for example, the transaction is out of Gas, then all the side effects are reverted but the fee is lost. Thus, ensuring enough gas is very important for initiating a transaction, especially for those sending Ether transactions.

Consensus protocol is employed to maintain a unique state of Ethereum in a certain time period. The consensus protocol of Ethereum is based on Proof of Work (PoW), similar to Bitcoin. This means that the Ethereum system will be safe unless a particular attacker owning 51% computing power of the whole network. Trust is built among Ethereum users for its transparent code execution process.

2.2 A Source Code Snippet of a Smart Ponzi Scheme

As mentioned above, smart contracts are usually written in high-level languages. *Solidity* is a contract-oriented, high-level language whose syntax is similar to that of JavaScript. *Solidity* is an important language in the Ethereum platform which running on EVM.

To understand how to identify a smart Ponzi scheme from its source code, we present a simplified smart contract written with solidity (Listing 1). The code snippet is extracted from Rubixi, left only the key code for understanding why it is a smart Ponzi scheme. Generally speaking, a smart contract consists of two parts: functions and data. Functions can be called by transactions or messages from other accounts or contracts. During the execution of a function, the data of that contract can be renewed.

The codes from line 2 to 11 of Listing 1 are data definition, which are used to describe the current state of the contract. For example, *balance* records the current balance of the smart contract and the structure *Participant* records the investor's address and payment. The function Rubixi in line 14 is the constructor which 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 function.* It is executed when an account sends Ether to the contract without data. Thus, when a participant invests Ether to the contract, the function *addPayout*, defined in line 19, is triggered. This function is the key of the contract as it implements the main logic of a Ponzi scheme: 1) records the address and payment of the investor (line21–22); 2) calculates fees (line 28); and 3) pays to previous investors when 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 payment (*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

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

[3] An address is a string of digits and characters that can be shared with anyone who wants to interact with the account. For example, the address of the aforementioned smart Ponzi scheme Rubixi is 0xe82719202e5965Cf5D9B6673B7503a3b92DE20be.

higher profit for them. It is worth mentioning that *pyramidMultiplier* should be set to above 100 for the promised profit of participants.

Taking fees from participants is the main purpose of operating a Ponzi scheme. It can be seen clearly from the code snippet that Rubixi charges every investment 10 percent, and the fees are collected by calling the function *collectAllFees* in line 37. 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. This piece of code clearly shows the logic of the contract payment, which obviously can be identified as a Ponzi scheme.

As seen from the code snippet, two kinds of transactions may occur to a smart contract: normal transactions and fried transactions. These transactions are publicly available and can be downloaded from ethereum.io[4] using the provided API.

Listing 1: The Simplified Rubixi

```
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 }
```

2.3 Deploy a Contract

As mentioned above, an Ethereum contact is a series of "Ethereum virtual machine code" or "EVM code" residing in the Ethereum blockchain. We call this *bytecode* of a contract. In order to write smart contracts conveniently, a high-level language is used (e.g., the *Solidity* language). Thus, to deploy a contract, the first thing to do is to compile the source code into EVM bytecode. The EVM bytecode is composed of a series of bytes. Each byte is an operation. For human readable, each operation corresponds with a mnemonic form. For example, the mnemonic form of EVM bytecode 0x10 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 [40] contains a complete list of the EVM bytecode and its mnemonic form, i.e., opcode. A disassembler[5] can be used to get the operation code of a contract from bytecode. Operation code consist of a series of *opcode* and *operand*. For example, the first 5 rows of the operation code in Rubixi are: *PUSH1 0x60; PUSH1 0x40; MSTORE; CALLDATASIZE; ISZERO*. *PUSH1* is an opcode and *0x40* is an operand.

To make the contract callable from other accounts, the bytecode of a smart contract should be deployed in the main Ethereum network. A special transaction targeted to the zero-account (the account with the address 0) creates a new contract. The bytecode of the contract provides as the payload of that transaction and will be executed; the result will be stored in the code field of the new contract account and be record permanently on the blockchain until the contract being killed by the creator. The address of the new contract will return to the creator, which can then be shared with the others.

3 DATA, FEATURE EXTRACTION AND CLASSIFICATION MODEL

In order to establish an effective model to detect smart Ponzi schemes, 1382 verified smart contracts were collected from the Website *http://etherscan.io*. Reported by a previous study [3] these smart contracts were inspected manually to check whether it is a smart Ponzi scheme or not. We recheck the results and tidy them as ground truth data in our model. Specifically, 131 Ponzi scheme contracts and 1251 non-Ponzi scheme contracts were collected. To establish the model, we downloaded the corresponding data and extracted two categories of features from the data. This section provides an overview of the data and features. The feature extraction method is also introduced.

3.1 Data

As seen from Fig. 2, two kinds of data including transactions and source codes were collected. The transaction data contain normal transactions and fired transactions in form of JSON files. All these data were collected through the APIs provided by etherscan.io[6]. Please visit our website[7] for more detailed information. It is worth noting that only the last 10000 transactions can be download due to the limitation of ethescan.io. However, this limitation has only a little influence to our study and was ignored.

The source code of a contract is very important for detecting its function. However, open source code is not compulsory, though it is suggested for public inspection on the Ethereum platform. In order to establish a practical model that can be used to detect latent smart Ponzi schemes, we rely on only bytecode, which is publicly available for any contract. We first compile source code by using

[4] https://etherscan.io/apis#accounts

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

[6] https://etherscan.io/apis#accounts

[7] ibase.site/scamedb

the Ethereum local client to get the bytecode. Then, a disassembly tool is used to get the operation code. Finally, we extract all the opcodes and calculated their frequency in a contract.

3.2 Account Features

Due to the fraudulent behavior, Ponzi schemes have several unique features compared with normal organizations. There are at least three characteristics in Ponzi contracts: 1) these contracts usually send Ether to accounts once investing to the contract; 2) some accounts receive more counts of payment than investment, For example, the creator who charge fees frequently from the contract; and 3) the balance of the contract may be low, as a Ponzi scheme is always trying to maintain an image of fast and high return.

The intrinsic characteristics of a Ponzi scheme determine its behavior, which can be used to judge whether it is a Ponzi scheme. The Ether flow of a contract is a good representation of such behavior. To show the behavioral characteristics, we introduce *Ether flow graph* of a contract, drawing by using the corresponding transactions. Unlike commonly used cash flow graph, an Ether flow graph is used to display transactions between the contract and its participants. The transactions have two directions: the participant either sends or receives Ether from the contract. We denote the first direction as investment transaction and the second as payment transaction of the participant. The investment transitions are denoted by red circles and the payment transactions are denoted by green triangles in the graph. The x-axis represents the time line, while the y-axis represents individual participants. By reading the graph, we can easily see the transactions that each participant are involved along the time line. The corresponding Ether amounts in the transactions are reflected by the size of the circles or triangles. The transactions between the contract and a participant are arranged in a horizontal line in order of the transactions' timestamps. Participants are ordered by the timestamp of their first transaction with the contract. Thus, the 0th participant is usually the creator of the contract. In general, an investment transaction should be followed by a payment transaction in normal economic activity, but it is not true in a smart Ponzi scheme.

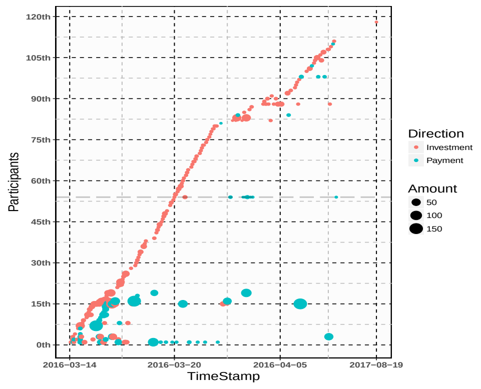


Figure 3: The Ether Flow Graph of Rubixi (A Typical Ponzi Scheme Contract)

Fig. 3 and 4 display the Ether flow graphs of two contracts: Rubixi and LooneyLottery. Rubixi is a typical smart Ponzi scheme and LooneyLottery[8] is a normal lottery game contract. The Ether flow graph of Rubixi contract (Fig. 3) shows that the creator of the contract and its very early participants are winners of the game. There are almost 120 participants involved in the contract, but most payments pertain to the first 25 lucky participants. Notably, there is an abnormal behavior pertaining to the 54th participant (presented as a gray dashed line in the graph), as he or she sent only twice but received payments many times. This abnormal behavior occurs because there is a bug in the source code which can be used to change the owner of the contract and collect fees [2].

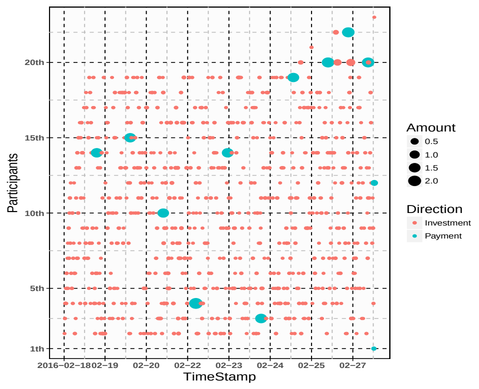


Figure 4: The Ether Flow Graph of LooneyLottery (A Typical Non-Ponzi Scheme Contract)

A significant difference can be found between the two figures: Fig. 3 contains more participants, but many participants interact less with the contract as compared with Fig. 4. From the payment perspective, each contract has relatively few payments, however, most payments in Fig. 3 pertain to the anterior participants but payments in Fig. 4 exhibit more randomness. These differences can be easily derived by the function of the contract. Thus, in turn, the account behavior can be used to classify smart contracts.

Through inspecting the Ether flow graph of Rubixi, we can find that: 1) the payment transactions usually occurs after an investment transaction, which indicates that the contract usually pays to known accounts; 2) many investment transactions have no followed payment transactions; and 3) some participants have more payment transactions than investment transactions. Based on these observations and characteristics: we extract seven key features in contracts as follows:

- *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.
- *Balance (Bal):* the balance of the smart contract.
- *N_investment (N_Inv) :* the number of investments.
- *N_payment (N_pay) :* the number of payments.
- *Difference index (D_ind) :* this index is used to measure the difference of counts between payment and investment for all participants in a contract. Suppose that there are p participants pertaining to the contract, v is a vector with length p,

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

m_i *and* n_i denote the counts of investments and payments of the ith participant. To calculate the difference index, each element of the vector v is first computed by $v[i] = n_i - m_i$, then

$$D\_ind = \begin{cases} 0 & if\ v = 0\ or\ p \leq 2; \\ s & otherwise \end{cases}$$

where s is the skewness of vector v. For a smart Ponzi scheme contract, D_ind is usually negative, as many participants invest more and receive less.

- *Paid rate (Pr):* the proportion of investors who received at least one payment.
- *N_maxpay (N_max):* the maximum of counts of payments to participants.

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.

Table 1: Statistics of Extracted Account Features

	Ponzi Scheme contracts						
	Kr	Bal	N_inv	N_pay	D_ind	Pr	N_max
Mean	0.89	4.65	56.84	92.49	-1.04	0.62	36.12
Median	1.00	0.26	17.00	21.00	-0.65	0.66	11.00
Sd	0.29	15.51	119.41	204.71	1.95	0.30	94.36
	Non-Ponzi Scheme Contracts						
	Kr	Bal	N_inv	N_pay	D_ind	Pr	N_max
Mean	0.49	22319.60	653.44	540.74	-0.51	0.43	237.95
Median	0.50	0.10	6.50	4.00	0.00	0.40	2.00
Sd	0.43	187549.23	3986.45	2195.42	6.05	0.41	1095.08

As seen clearly from the table, the statistics between Ponzi scheme contracts and non-Ponzi scheme contracts are hugely different. For example, the median of Known rate (Kr) of Ponzi scheme contract is 1 but only 0.5 for non-Ponzi scheme contracts. The high Kr of Ponzi scheme contract shows that many contracts pay to those once invested it, which is a significant feature for Ponzi scheme contract. As for balance (Bal), the difference between the medians of the two scheme is minor, but the difference of standard deviations (Sd) is very large. The low standard deviation of Ponzi scheme contracts indicating that many contracts have relatively low balance. On the other hand, the high standard deviation of non-Ponzi scheme contracts indicates that some contracts have very large balance. Meanwhile, as the median of non-Ponzi scheme contracts is only 0.1, which means that half of non-Ponzi scheme contracts have a balance less than 0.1. As a matter of fact, many non-Ponzi scheme contracts have zero balance.

3.3 Code Features

Opcodes are successful in analyzing the latent problem of smart contracts as it reflects the logic of smart contracts from the aspect of Ethereum Virtual Machine (EVM) [2, 6]. We expect that features extracted from opcodes are useful in detecting latent smart Ponzi schemes. We extracted all the opcodes and calculated their frequency. 64 different opcodes are found in the 1382 contracts' operation codes.

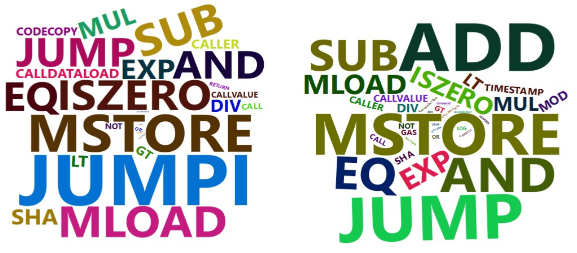


Figure 5: The Opcode Cloud of Rubixi (left) and LooneyLottery (right)

Fig. 5 shows the cloud graphs of opcodes in the aforementioned two smart contracts. Three most frequent opcodes, PUSH, DUP and SWAP are removed to make the graphs more easily seen. This is because EVM is stack-based, these opcodes appear frequently in every contract.

Although it is impossible to identify the type of smart contract just by observing the cloud graph of opcode, it is easy to see that the two smart contracts are obviously different. Intuitively, there are at least two significant differences: Rubixi contains more judgments and LooneyLottery contains more randomness. Actually, the first difference can be seen clearly from the graph, the Rubixi contract contains relatively more JUMPI (7.8%), and the LooneyLottery contract contains more JUMP (2.6%). The difference between these two opcodes is that the former is a conditional version of the latter. To detect the second difference needs more observation. The LooneyLottery contract contains 4 TIMESTAMPs while the Rubixi contains none. The reason is that the opcode TIMESTAMP is used to get the block's timestamp, which is a commonly used as a random variable of the system. The above analysis indicates that opcode features may be feasible in detecting Ponzi scheme contracts.

3.4 Classification Model

In order to distinguish Ponzi scheme contracts from other smart contracts with high accuracy and low false positives, we use XGBoost, which is one of the most popular machine learning algorithms and is proved to be a good method in many problems [5]. This subsection provides a simple introduction of XGBoost and its parameters.

XGBoost is short for "Extreme Gradient Boosting" and is an improved version of gradient tree boosting [13][9]. Unlike GBM, XGBoost introduced two important improvements: regularization and tree pruning. Basically, XGBoost provides a regularization in the objective function, which is used to avoid overfitting in the tree-based model. Meanwhile, it changed the tree pruning method from stopping when a negative loss encountered to the post-pruning.

Specifically, 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

[9]Gradient tree boosting is also known as gradient boosting machine (GBM)

is a verified Ponzi scheme contract. We use XGBoost aiming to minimize the following objective function:

$$Obj(\theta) = L(\theta) + \Omega(\theta),$$

where L is the training loss function and Ω is the regulation term. The training loss function measures how predictive the model is on the training data and the regulation term penalizes the complexity of the model, which helps to avoid overfitting.

Unlike traditional classification model which directly returns the class label, we train a logistic regression model for binary classification, which outputs a probability. Any contract with the predicted probability larger than 0.5 is considered as a Ponzi scheme contract. The corresponding logistic loss function is as follows:

$$L(\theta) = \sum_i [y_i \ln(1 + e^{-\hat{y}_i}) + (1 - y_i)\ln(1 + e^{\hat{y}_i})]$$

As for the regulation term, the first thing to do is redefining a tree that is convenient to measure the complexity. In XGBoost, a tree is a function that maps an instance to a leaf weight. Specifically,

$$f(x) = \omega_{q(x)}, \ \ \omega \in R^T, \ q \ : R^d \rightarrow \ \{1, 2, \dots, T\},$$

where ω is the leaf weight of the tree, T is the number of leafs and q stands for the tree structure. The complexity of the tree is defined as

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} \omega_j^2,$$

where γ and λ are parameters of the model.

To ensemble trees, the objective function is rewritten as follows

$$Obj = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k),$$

where $\hat{y}_i = \sum_{k=1}^{K} f_k(x_i)$, K is the number of trees.

For the learning process, XGBoost introduces *additive training*, which starts from constant prediction and adds new function each time as follows:

$$\begin{array}{rcl} \hat{y}_i^{(0)} & = & 0 \\ \hat{y}_i^{(1)} & = & f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i) \\ \hat{y}_i^{(2)} & = & f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i) \\ & \dots & \\ \hat{y}_i^{(t)} & = & \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i). \end{array}$$

By applying the above equations, the objective function can be written as

$$Obj^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) + constant,$$

and XGBoost aims to find the best f_t to minimize this function at each step. To grow the tree, it tries to add the best split of a feature that maximizes *gain* on each leaf nodes. XGBoost adopts the post-pruning method, growing a tree to maximum depth and then recursively pruning all the leaf splits with negative gain.

4 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 based on the comparison of the two categories of extracted features are summarized. Finally, we analyze the importance of features.

4.1 Experiment Setting

Datasets. In order to compare the discriminative power of the two categories of features, we conduct experiments on three kinds of features: account, opcode and their combination. For all the three experiments, we first adopt 5-fold cross-validation to find the best parameters of the model and then we split the corresponding data into 80% for training and 20% for testing and conduct the experiment for ten times by using the best parameters found. The average results are summarized in Table 2.

Evaluation metrics. Different from the commonly used metrics of error rate in classification problems, we use three metrics, precision, recall, and F-score, to evaluate the performances of the model. Here is a brief introduction:

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

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

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

4.2 Results Summary

Table 2 summarizes the performances of the three kinds of features in detecting Smart Ponzi schemes.

Table 2: A Feature Performannce Comparison

Features	Precision	Recall	F-score
Account	0.74	0.32	0.44
Opcode	0.90	0.80	0.84
Account + Opcode	0.94	0.81	0.86

Several conclusions can be made from the table. First, the account features, to our surprise, are not efficient in detecting smart Ponzi scheme. We expected that the account features would have a good performance, as smart Ponzi schemes behave differently. However, the low recall shows that the model based on these features is almost useless. In contrast, the opcode features are very efficient as expected. The possible reason of this result may be that many smart contracts are experimental, which makes it hard to detect their types from behavior. In fact, many smart contracts have no transactions. Another possible reason is that the number of account features is too few. Second, the corresponding metrics show that modes based on just opcode features can be used for detecting smart Ponzi schemes. Finally, the performance of the model can be improved by combining opcode features with account features.

4.3 Feature Analysis

To further understand the discriminative power of the extracted features, we display the ten most significant features in Fig. 6. The description of opcodes in the graph are summarized in Table 3.

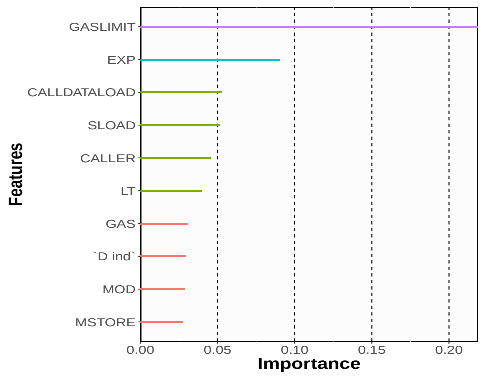


Figure 6: The Importance of the Ten Most Significant Features

It can be seen clearly from the graph that the most significant feature is GASLIMIT, which is used to get the block's gas limits. To better understand why GASLIMIT is the most significant feature, we first calculate the number of contracts that have the opcode and find that 57% of the Ponzi scheme contract has it compared with only 4% in non-Ponzi scheme contracts. Thus smart Ponzi contracts are more referring to the block's gas limits as compared with other contracts. However, this result is against common sense because getting the block's gas limit is completely unnecessary for a smart Ponzi scheme. In order to find the real reason, we selected some source codes of Ponzi scheme contracts which have GASLIMIT in their operation codes. We hope to find out what source code brings this. We find that these contracts import oracle APIs which results in the opcode.

Table 3: Opcode and its Description

Opcode	Description
GASLIMIT	Get the block's gas limit.
EXP	Exponential operation.
CALLDATALOAD	Get input data of current environment.
SLOAD	Load word from storage.
CALLER	Get caller address.
LT	Less-than comparision.
GAS	Get the amount of available gas.
MOD	Modulo remainder operation.
MSTORE	Save word to memory.

Another worth mentioning opcode is CALLER, which is used to get the caller address. Remember, smart Ponzi schemes need to record those investors. Majority of the rest of the opcodes are arithmetic-operation-related. Generally speaking, those opcodes seem to indicate a class of special contracts that have more payment transactions, need to remember caller and have many arithmetic operations. Thus, Ponzi scheme contract belongs to it.

Only one account feature (D_ind) is found in the ten most important features. This is not surprising, since the difference index is a discriminative account feature as discussed in Section 3.2.

4.4 Application

To estimate the number of smart Ponzi schemes on Ethereum, the previous study detects hidden smart Ponzi schemes (Ponzi scheme contracts without source codes) by using similarity between two bytecode files [3]. 54 hidden smart Ponzi schemes were found in the paper. To verify the reliability of our model as compared with it, we predict the 54 hidden contracts with the model based on code features. The result shows that 45 out of 54 (83%) contracts are smart Ponzi schemes. In order to understand why the remainder 9 contracts failed to be detected as Ponzi scheme contracts by our model, we manually check all their transactions. The findings are as follows:

- Two contracts actually have source codes[10]. They refer to the same project on github[11]. By reading the introduction page and checking the source codes, one can find that it is not a smart Ponzi scheme but a kind of game. Our model successfully excludes it.
- A contract[12] interacted with only the creator. Though we can not say that it is not a smart Ponzi scheme, but it seems more like a test-transaction-contract from the transaction records. Another contract[13], which have 5 successful transactions but only one payment to an unacquainted address, is possibly not a smart Ponzi scheme as the balance is relatively large and the transaction pattern is not conformed to any kind of smart Ponzi schemes summarized in [3].
- A contract[14], which has only one transaction with amount large than 0 (including normal and fired transactions) and all the fired transactions are function call from unknown accounts, cannot be a smart Ponzi scheme. A similar situation can be found in another contract[15].
- The order of payments in the two contracts[16] are strange. There are only a few participants, but anterior participants receive payments later than posteriors, which indicates that they are probably not smart Ponzi schemes.
- The last contract[17], which has similar behavior with smart Ponzi schemes but more payments than investments, so it is hard to drawn a conclusion.

These findings suggest that our model is more accurate in detecting smart Ponzi schemes compared with bytecodes similarity.

To estimate how many smart Ponzi schemes on Ethereum, we first downloaded all the contracts created before May 7, 2017 (the

[10]0xb56e95aea830b0242be6a5d0239ed7f71408563b
0x0abce3be0075d067e12da8d266de752e20ff9842
[11] https://github.com/rolandkofler/matthew
[12] 0x79280ded572a0a7dfd31dfcc5baef3121ef0fee6
[13]0xd361e374be9e3907fceac60c6ea5cbdce89fc9ae
[14]0x0d5919572552c6c8c752aa402bd033f2b2886bbc
[15]0xa820487e57656771b21ab533cb99e8d347aa20ef
[16]0x5158cf97c3e001b402ccb0f9063736ee8d6dad5a
0xf52ecc525d998eb880911a268b0fa4bc7d69a435
[17]0xa1c1983aa3599657a74cf5a563e880eedc57ae4f

same date range as in [3]). We obtained 280704 contracts in total. Then we extracted all the opcode features of these smart contracts and predicted with our model. 386 smart Ponzi schemes (include these verified) were found by the model. Thus it is estimated that almost 434 ($386 \times precision/recall$) smart Ponzi schemes run on Ethereum platform before May 7, 2017, accounting for 0.15% of all the contracts. Fig. 7 shows the counts of detected smart Ponzi schemes with corresponding scores (probabilities). It can be seen clearly from the graph that much more detected smart Ponzi schemes have relatively high scores, indicating that the problem of Ponzi schemes is more serious than estimated. As a matter of fact, only 191 smart Ponzi schemes reported in the recent study [3], which is much less than the number estimated here.

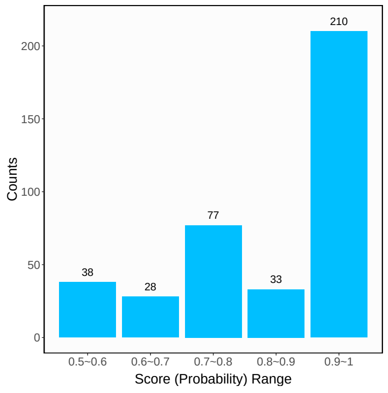


Figure 7: The Number of Detected Smart Ponzi Schemes and Scores (Probability)

5 RELATED WORK

Since the creation of bitcoin, blockchain technology became a research hotspot. Three types of research can be found in the literature. The first type focuses on the underlying mechanism. Many consensus mechanisms were proposed, such as proof of stake (PoS) [17], practical byzantine fault tolerance (PBFT) [14, 21] and ripple [34]. Furthermore, some research focused on improving the existing mechanism [11]. The second type discusses the application of blockchain technology. Since blockchain technology has many favorable characteristics, many applications can be found in finance service [20, 31], Internet of things (IoT) [7, 10],and social services [28]. The last but the most related type of work is data mining on blockchain. Thank to the public accessible characteristic, blockchain provides an unprecedented opportunity for data analysis to answer various questions, for example, usage characteristic [22, 33, 41] , anonymity [1, 32] and economic behavior analysis[18, 19]. More imforamtion can be found in the servey [44].

With the development of the Internet, online “High-yield” investment program (HYIP) become a typical form of Ponzi schemes. A preliminary analysis was provided on economic aspects of it by using data collected from HYIP websites [24]. More detailed research was conducted in [29], where a model was set out to estimate the turnover and profits of HYIPs. Both papers focused on HYIPs which use centralized virtual currencies. The creation of blockchain technology makes cryptocurrency an ideal currency for scammers. In blockchain, the ledger records every transaction and these transactions can be accessed publicly. Valuable data can thus be obtained by researchers to investigate scams on blockchain. Among different scams, bitcoin is the most notable one[8, 23, 26]. Marie Vasek and Tyler Moore provided an empirical analysis of Bitcoin-based scams [38]. They found 32 HYIPs and estimated more than 4 million USD involved. A recent study focused on the economic aspects of smart Ponzi schemes [3], they use *normalized Levenshtein distance* [42] as a measure of similarity between bytecodes to detect hidden smart Ponzi schemes. Different from their study, this paper focused on finding a verifiable method to detect smart Ponzi schemes in bytecodes.

6 CONCLUSION AND FUTURE WORK

Financial scams based on blockchain and cryptocurrency has become an important research problem. With the development of blockchain technology, Ponzi scheme is now under the veil of smart contracts. In this study, we propose an approach to detect smart Ponzi schemes. Using the manually checked samples and XGBoost, a regression tree model based on extracted account features and code features is build. The experimental results indicate that the proposed model has high accuracy and can be used to detect latent smart Ponzi schemes in practice. The most significance results is that using our extracted code features that are publicly accessible in any running contract, is enough to build a practical model for detecting Ponzi scheme contract at the moment of its creation. In addition, we estimate that there are more than 400 smart Ponzi schemes on Ethereum, which are far more than the previous estimation.

In the future, we are going to further this study in three aspects. Firstly, to extend the ground truth data and improve the classification model. As the number of smart contracts having source codes keep increasing every day, it is possible to extend the ground truth data by manually checking the source codes. With more ground truth data, more accurate and credible model can be developed. Secondly, to build a unified platform to evaluate and score every smart contract for every possible scam. We have noticed that there is an open source project on Github to keep track of all the current Ethereum scams[18]. However, the website collects data by manual user reports and combines them. We are trying to establish a similar website focusing on detecting smart contracts by data mining methods. Thirdly, to study other kinds of scams. Except for smart Ponzi scheme, many other scams are taking the advantage of blockchain technology. It is necessary to study this question to promote the development of blockchain technology.

ACKNOWLEDGMENTS

The work described in this paper was supported by the the National Key Research and Development Program (2016YFB1000101), the National Natural Science Foundation of China (61722214, 61472338, 61773410), the Program for Guangdong Introducing Innovative and Enterpreneurial Teams (2016ZT06D211) ,the Pearl River S&T Nova Program of Guangzhou (201710010046). This collaboration between China and Sweden has also been supported by STINT initiation grant for international collaboration (grant number: IB2017-6978). Zibin Zheng is the corresponding author.

[18]https://etherscamdb.info/

REFERENCES

[1] Elli Androulaki, Ghassan Karame, Marc Roeschlin, Tobias Scherer, and Srdjan Capkun. 2013. Evaluating User Privacy in Bitcoin. In *Proceedings of International Conference on Financial Cryptography and Data Security*. Springer, 34–51.

[2] Nicola Atzei, Massimo Bartoletti, and Tiziana Cimoli. 2017. A Survey of Attacks on Ethereum Smart Contracts (SoK). In *Proceedings of International Conference on Principles of Security and Trust*. Springer, 164–186.

[3] Massimo Bartoletti, Salvatore Carta, Tiziana Cimoli, and Roberto Saia. 2017. Dissecting Ponzi schemes on Ethereum: Identification, analysis, and impact. (2017). arXiv:1703.03779

[4] Vitalik Buterin et al. 2014. A next-generation smart contract and decentralized application platform. *Ethereum white paper* (2014).

[5] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A scalable tree boosting system. In *Proceedings of International Conference on Knowledge Discovery and Data Mining*. ACM, 785–794.

[6] Ting Chen, Xiaoqi Li, Xiapu Luo, and Xiaosong Zhang. 2017. Under-optimized smart contracts devour your money. In *Proceedings of International Conference on Software Analysis, Evolution and Reengineering*. IEEE Computer Society, 442–446.

[7] Konstantinos Christidis and Michael Devetsikiotis. 2016. Blockchains and smart contracts for the internet of things. *IEEE Access* 4 (2016), 2292–2303.

[8] Nicolas Christin. 2013. Traveling the silk road: A measurement analysis of a large anonymous online marketplace. In *Proceedings of International World Wide Web Conference*. International World Wide Web Conferences Steering Committee / ACM, 213–224.

[9] CoinDesk. 2017. Understanding Ethereum-blockchain research report. (2017). Retrieved October 1, 2017 from www.coindesk.com/research/understanding-ethereum-report/

[10] Marco Conoscenti, Antonio Vetro, and Juan Carlos De Martin. 2016. Blockchain for the Internet of Things: A systematic literature review. In *Proceedings of International Conference of Computer Systems and Applications*. IEEE, 1–6.

[11] Christian Decker, Jochen Seidel, and Roger Wattenhofer. 2016. Bitcoin meets strong consistency. In *Proceedings of International Conference on Distributed Computing and Networking*. ACM, 13:1–13:10.

[12] Craig Kent Elwell, M Maureen Murphy, and Michael V Seitzinger. 2013. Bitcoin: Questions, answers, and analysis of legal issues. (2013). Retrieved October 1, 2017 from https://digital.library.unt.edu/ark:/67531/metadc272070/

[13] Jerome H Friedman. 2001. Greedy function approximation: A gradient boosting machine. *Annals of statistics* (2001), 1189–1232.

[14] Hyperledger. 2015. Hyperledger project. (2015). Retrieved October 1, 2017 from https://www.hyperledger.org/

[15] Ari Juels, Ahmed Kosba, and Elaine Shi. 2016. The ring of Gyges: Investigating the future of criminal smart contracts. In *Proceedings of International Conference on Computer and Communications Security*. ACM, 283–295.

[16] Garrett Keirns. 2017. "Gemcoin" Ponzi scheme operator hit with $74 million judgment. (March 2017). Retrieved October 1, 2017 from https://www.coindesk.com/gemcoin-ponzi-scheme-operator-hit-74-million-judgment/

[17] Sunny King and Scott Nadal. 2012. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. *Ppcoin white paper* (2012).

[18] Dániel Kondor, István Csabai, János Szüle, Márton Pósfai, and Gábor Vattay. 2014. Inferring the interplay between network structure and market effects in Bitcoin. *New Journal of Physics* 16, 12 (2014), 125003.

[19] Dániel Kondor, Márton Pósfai, István Csabai, and Gábor Vattay. 2014. Do the rich get richer? An empirical analysis of the Bitcoin transaction network. *PloS one* 9, 2 (2014), e86197.

[20] Dániel Kondor, Márton Pósfai, István Csabai, and Gábor Vattay. 2016. From "Blockchain Hype" to a Real Business Case for Financial Markets. *Available at SSRN 2760184* (2016).

[21] David Mazieres. 2015. The stellar consensus protocol: A federated model for internet-level consensus. *Stellar Development Foundation* (2015).

[22] Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Kirill Levchenko, Damon McCoy, Geoffrey M. Voelker, and Stefan Savage. 2016. A fistful of Bitcoins: characterizing payments among men with no names. *Commun. ACM* 59, 4 (2016), 86–93.

[23] Tyler Moore and Nicolas Christin. 2013. Beware the Middleman: Empirical Analysis of Bitcoin-Exchange Risk. In *Proceedings of International Conference on Financial Cryptography and Data Security*. Springer, 25–33.

[24] Tyler Moore, Jie Han, and Richard Clayton. 2012. The Postmodern Ponzi Scheme: Empirical Analysis of High-Yield Investment Programs. In *Proceedings of International Conference on Financial Cryptography and Data Security*, Vol. 7397. Springer, 41–56.

[25] DAVID Z. MORRIS. 2017. The rise of cryptocurrency Ponzi schemes. (May 2017). Retrieved October 1, 2017 from https://www.theatlantic.com/technology/archive/2017/05/cryptocurrency-ponzi-schemes/528624/

[26] Malte Moser, Rainer Bohme, and Dominic Breuker. 2013. An inquiry into money laundering tools in the Bitcoin ecosystem. In *Proceedings of eCrime Researchers Summit (eCRS)*. IEEE, 1–14.

[27] Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system. (2008).

[28] Namecoin. 2014. Namecoin project. (2014). Retrieved October 1, 2017 from https://www.namecoin.org/

[29] Jens Neisius and Richard Clayton. 2014. Orchestrated crime: The high yield investment fraud ecosystem. In *proceedings of APWG Symposium on Electronic Crime Research (eCrime)*. IEEE, 48–58.

[30] Alex Norta. 2015. Creation of Smart-Contracting Collaborations for Decentralized Autonomous Organizations. In *Proceedings of International Conference on Perspectives in Business Informatics Research*. Springer, 3–17.

[31] Gareth William Peters and Efstathios Panayi. 2015. Understanding modern banking ledgers through blockchain technologies: future of transaction processing and smart contracts on the internet of money. (2015). arXiv:1511.05740

[32] Fergal Reid and Martin Harrigan. 2013. An analysis of anonymity in the bitcoin system. In *Security and privacy in social networks*. Springer, 197–223.

[33] Dorit Ron and Adi Shamir. 2013. Quantitative Analysis of the Full Bitcoin Transaction Graph. In *Proceedings of International Conference on Financial Cryptography and Data Security*. Springer, 6–24.

[34] David Schwartz, Noah Youngs, and Arthur Britto. 2014. The Ripple protocol consensus algorithm. *Ripple Labs Inc White Paper* (2014).

[35] Stan Higgins. 2015. SEC seizes assets from alleged altcoin pyramid scheme. https://www.coindesk.com/sec-seizes-alleged-altcoin-pyramid-scheme/. (Oct. 2015).

[36] Melanie Swan. 2015. *Blockchain: Blueprint for a new economy*. O'Reilly Media, Inc.

[37] Nick Szabo. 1996. Smart contracts: Building blocks for digital markets. (Sept. 1996). Retrieved October 1, 2017 from http://www.fon.hum.uva.nl/

[38] Marie Vasek and Tyler Moore. 2015. There's no free lunch, even using Bitcoin: Tracking the popularity and profits of virtual currency scams. In *Proceedings of International Conference on Financial Cryptography and Data Security*. Springer, 44–61.

[39] Wikipedia. 2017. Ponzi scheme. (Oct. 2017). Retrieved October 1, 2017 from https://en.wikipedia.org/wiki/Ponzi_scheme

[40] Gavin Wood. 2014. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Yellow Paper* (2014).

[41] Aaron Yelowitz and Matthew Wilson. 2015. Characteristics of Bitcoin users: an analysis of Google search data. *Applied Economics Letters* 22, 13 (2015), 1030–1036.

[42] Li Yujian and Liu Bo. 2007. A normalized Levenshtein distance metric. *IEEE transactions on pattern analysis and machine intelligence* 29, 6 (2007), 1091–1095.

[43] Zibin Zheng, Shaoan Xie, Hongning Dai, Xiangping Chen, and Huaimin Wang. 2016. Blockchain challenges and opportunities: A survey. *International Journal of Web and Grid Services* (2016).

[44] Zibin Zheng, Shaoan Xie, Hongning Dai, Xiangping Chen, and Huaimin Wang. 2017. An overview of blockchain technology: Architecture, consensus, and future trends. In *Proceedings of International Congress on Big Data*. IEEE Computer Society, 557–564.