4-2023

Code will tell: Visual identification of Ponzi schemes on Ethereum

Xiaolin WEN
Sichuan University

Kim Siang YEO
Singapore Management University, ks.yeo.2021@mitb.smu.edu.sg

Yong WANG
Singapore Management University, yongwang@smu.edu.sg

Ling CHENG
Singapore Management University, lingcheng.2020@phdcs.smu.edu.sg

Feida ZHU
Singapore Management University, fdzhu@smu.edu.sg

See next page for additional authors

Citation

WEN, Xiaolin; YEO, Kim Siang; WANG, Yong; CHENG, Ling; ZHU, Feida; and ZHU, Min. Code will tell: Visual identification of Ponzi schemes on Ethereum. (2023). *CHI'23: Extended Abstract Proceedings, CHI Conference on Human Factors in Computing Systems, Hamburg, Germany, April 23-28*. 1-6.
Available at: https://ink.library.smu.edu.sg/sis_research/8577

Author

Xiaolin WEN, Kim Siang YEO, Yong WANG, Ling CHENG, Feida ZHU, and Min ZHU

Code Will Tell: Visual Identification of Ponzi Schemes on Ethereum

Xiaolin Wen
Sichuan University
Chengdu, China
Singapore Management University
Singapore
wenxiaolin@stu.scu.edu.cn

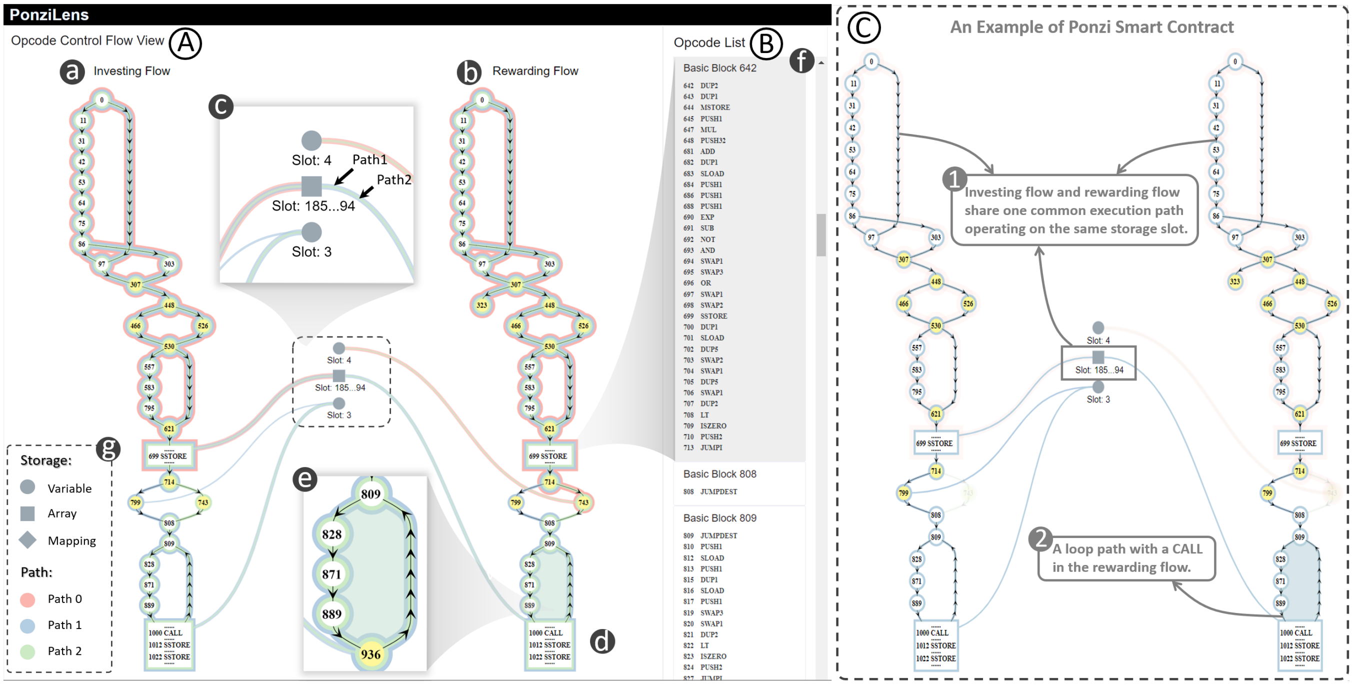
Kim Siang Yeo
Singapore Management University
Singapore
ks.yeo.2021@mitb.smu.edu.sg

Yong Wang*
Singapore Management University
Singapore
yongwang@smu.edu.sg

Ling Cheng
Singapore Management University
Singapore
lingcheng.2020@phdcs.smu.edu.sg

Feida Zhu
Singapore Management University
Singapore
fdzhu@smu.edu.sg

Min Zhu
Sichuan University
Chengdu, China
zhumin@scu.edu.cn



Basic Block 642

```
642  DUP2
643  DUP1
644  MSTORE
645  PUSH1
647  MUL
648  PUSH32
681  ADD
682  DUP1
683  SLOAD
684  PUSH1
686  PUSH1
688  PUSH1
690  EXP
691  SUB
692  NOT
693  AND
694  SWAP1
695  SWAP3
696  OR
697  SWAP1
698  SWAP2
699  SSTORE
700  DUP1
701  SLOAD
702  DUP5
703  SWAP2
704  SWAP1
705  DUP5
706  SWAP1
707  DUP2
708  LT
709  ISZERO
710  PUSH2
713  JUMPI
```

Basic Block 808

```
808  JUMPDEST
```

Basic Block 809

```
809  JUMPDEST
810  PUSH1
812  SLOAD
813  PUSH1
815  DUP1
816  SLOAD
817  PUSH1
819  SWAP3
820  SWAP1
821  DUP2
822  LT
823  ISZERO
824  PUSH2
827  JUMPI
```

Investing flow and rewarding flow share one common execution path operating on the same storage slot.

A loop path with a CALL in the rewarding flow.

Figure 1: The user interface of *PonziLens* consists of (A) *Opcode Control Flow View* and (B) *Opcode List*. (A) The *Opcode Control Flow View* shows (a) investing flow, (b) rewarding flow, and (c) storage interactions — all of which are critical for identifying a Ponzi smart contract. (B) The *Opcode List* shows all the original operation codes of a smart contract, where the operation code of a basic block can be highlighted (f). (d) shows that a basic code block in control flow can be unfolded to check the critical instructions within it. (e) shows an execution loop within a CALL instruction. (g) is the legend illustrating the storage type and the aggregated paths. (C) shows the Opcode Control Flow View with the aggregated path in blue *Path1* highlighted.

ABSTRACT

Ethereum has become a popular blockchain with smart contracts for investors nowadays. Due to the decentralization and anonymity of Ethereum, Ponzi schemes have been easily deployed and caused significant losses to investors. However, there are still no explainable and effective methods to help investors easily identify Ponzi schemes and validate whether a smart contract is actually a Ponzi scheme. To fill the research gap, we propose *PonziLens*, a novel

[*] The corresponding author.

CHI EA '23, April 23–28, 2023, Hamburg, Germany
© 2023 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9422-2/23/04.
https://doi.org/10.1145/3544549.3585861

visualization approach to help investors achieve early identification of Ponzi schemes by investigating the operation codes of smart contracts. Specifically, we conduct symbolic execution of opcode and extract the control flow for investing and rewarding with critical opcode instructions. Then, an intuitive directed-graph based visualization is proposed to display the investing and rewarding flows and the crucial execution paths, enabling easy identification of Ponzi schemes on Ethereum. Two usage scenarios involving both Ponzi and non-Ponzi schemes demonstrate the effectiveness of *PonziLens*.

CCS CONCEPTS

• **Human-centered computing** → **Visual analytics**; **Information visualization**.

KEYWORDS

Ponzi scheme, visual identification, Ethereum, visual analytics

ACM Reference Format:
Xiaolin Wen, Kim Siang Yeo, Yong Wang, Ling Cheng, Feida Zhu, and Min Zhu. 2023. Code Will Tell: Visual Identification of Ponzi Schemes on Ethereum. In *Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems (CHI EA '23), April 23–28, 2023, Hamburg, Germany*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3544549.3585861

1 INTRODUCTION

With the prevalence of blockchain, Ethereum, a blockchain-based system, has become an increasingly popular way for investors to carry out decentralized, secure and anonymous transactions without an intermediate third party's credit endorsement [10, 29]. Ethereum incorporates smart contracts that define the transaction rules in the form of source code on the blockchain. Such smart contracts will be executed automatically on the Ethereum Virtual Machine (EVM) once the predefined conditions in the contract are met [6, 25]. All the transaction records and smart contract codes are publicly available and immutable on Ethereum.

Unfortunately, scammers have also leveraged the anonymity and immutability of blockchain and deployed various "trustworthy" frauds on Ethereum to cheat investors of their money, or Ether — the scarce digital money on Ethereum. Among all the frauds, Ponzi schemes [2, 19] are a popular investment scam on Ethereum that lure investors with a promise of high profits that are actually from the invested Ether of subsequent new investors, instead of actual investment appreciation income. Since all the byte codes of smart contracts on Ethereum are publicly available, it gives investors an illusion that smart contracts on Ethereum are credible, making them tend to trust the smart contracts. Also, due to the anonymity and immutability of blockchain, it is difficult to track and identify the fraudsters and also unable to revoke the Ponzi scheme transactions once the transactions are written into the blockchain. All these factors have made Ponzi schemes easy to be deployed on Ethereum and Ponzi schemes have caused significant economic losses to investors on Ethereum. According to Chen et al. [7], Ponzi schemes on Ethereum have led to losses of more than US$17 million by 2021.

Early studies have first leveraged transaction data for money flow analysis, and further detected Ponzi schemes on blockchain [5, 27, 28]. They intrinsically require that at least a group of investors have fallen into the trap of Ponzi schemes and cannot work for early detection of Ponzi schemes. More recent Ponzi scheme detection techniques [7, 12–14, 23, 31] have further investigated the operation code (opcode) of smart contracts on Ethereum. By considering the characteristics of opcodes (e.g., the operator frequency) of Ponzi schemes, they can achieve an early identification of Ponzi schemes before any investors are trapped by a Ponzi scheme. However, such techniques rely on holistic features of opcodes such as the operator frequency and cannot adapt to various Ponzi schemes to achieve a consistently high detection accuracy. Also, they totally ignore the semantic meaning of opcodes [7], making it difficult for investors to understand why a smart contract is predicted as a Ponzi scheme. An explainable and effective way to help investors identify Ponzi schemes on Ethereum is still missing.

In this paper, we fill the research gap by informing investors of the semantic meaning of the opcodes of smart contracts to facilitate Ponzi scheme identification on Ethereum. Specifically, we propose *PonziLens*, a visualization approach to show the investing and rewarding flows of smart contracts as well as their relations, revealing the essential characteristic of Ponzi schemes, i.e., *whether the reward of prior investors directly comes from the investments of subsequent new investors*. However, it is a challenging task due to the vast differences in the opcodes of various smart contracts as well as the difficulty of making common investors easily understand the function of complex opcodes. Inspired by prior studies [9, 16], we extract the Control Flow Graph from the opcode of a smart contract on Ethereum via symbolic execution, and further identify execution paths relevant to the investing and rewarding process of smart contracts by using crucial opcode instructions like *CALL, CALLER, SSTORE* and *SLOAD*. Some crucial features indicating a Ponzi scheme, including storage stacks shared by investing and rewarding flows and opcode loops, are also extracted. Then, we propose an intuitive directed-graph based visualization to show the investing and rewarding flows of smart contracts, where the crucial execution paths and common storage are also explicitly highlighted. *PonziLens* clearly visualizes all the Ether flows within a smart contract, and helps investors easily identify Ponzi schemes on Ethereum. To the best of our knowledge, it is the first time that the semantically-meaningful Ether flows of smart contracts have been visualized for Ponzi scheme identification. We showcase two usage scenarios, where both a Ponzi scheme and non-Ponzi scheme are investigated, to demonstrate the usefulness of *PonziLens*. In summary, the contributions of this paper can be summarized as follows:

- A novel visualization approach, *PonziLens*, to inform investors of the investing and rewarding flows of a smart contract and facilitate easy identification of Ponzi schemes on Ethereum.
- Two usage scenarios involving both Ponzi and non-Ponzi schemes to demonstrate the usefulness of our approach.

2 RELATED WORK

This paper is related to prior research on *automated detection of Ponzi schemes* and *visual analytics for blockchain data.*

Automated detection of Ponzi schemes Many approaches have been developed to achieve automated detection of Ponzi schemes in blockchain by analyzing the transaction data and the source code of smart contracts. For transaction-based approaches, they often leverage machine learning techniques, such as ordered boosting [12], attention neural networks [14] and behaviour forest [23], to learn the characteristics of Ponzi schemes and achieve automated Ponzi detection. These approaches intrinsically cannot work for early detection of Ponzi schemes before any transactions of a Ponzi scheme are invoked. For smart contract-based approaches, they mainly attempt to extract distinctive features of Ponzi schemes from smart contracts via different ways such as symbolic execution of opcode [7] and code attribution for de-anonymizing smart contracts [18]. Also, some recent works have combined transaction data with the source code of smart contract together for automated detection of Ponzi schemes [8, 13, 15, 31]. However, almost all the smart contract-based approaches rely on extracting high-level features and ignore the semantic meanings of the opcodes of smart contracts. It makes their result difficult to be understood by investors, which will be addressed in our approach.

Visual analytics for anomaly detection on blockchain Prior studies have developed visual analytics approaches to facilitate anomaly detection on blockchain. Transaction data capture all the interactions between entities, and has been explored for helping investors identify different anomalies on blockchain (especially bitcoin blockchain) in a general way. For example, Blockchain explorer [17], Biva [21], BitVis [24] and Bitconeview [11] provided detailed statistics of transactions on bitcoin blockchain, and also visualized the relations between different wallet addresses and transactions. These approaches can effectively reveal some anomalies such as money laundering and Ponzi schemes, as these anomalies show obvious characteristics in their transactions. Other visual analytics methods have also investigated the transaction data and highlighted some transaction features that are specific for one specific anomaly. For instance, Ahmed et al. [1] used taint tracking to trace the trail of stolen money back to its owner. Balthasar et al. [3] visualized the unique transaction patterns of money laundering such as the mixing of bitcoins, facilitating money laundering identification. Wen et al. [30] proposed NFTDisk, a visual analytic system, to help investors detect wash trading in NFT markets. Also, a few visualization approaches have also been proposed to analyze the source code of smart contracts for different purposes such as facilitating smart contract development [26] and Solidity code representation [22]. However, none of the above studies has attempted to visualize the source code of smart contracts for Ponzi scheme detection on Ethereum, which is the focus of this paper.

3 BACKGROUND

This section introduces the overall background of Ethereum and Ponzi schemes.

Data on Ethereum. The EVM is a *stack*-based architecture, where *stack* is an internal place to store temporal variables [29]. Besides *stack*, the EVM also stores data in two other places [7, 29]: *memory* and *storage*. *Memory* is a byte array storing the data for function execution and *storage* is used to permanently keep data on the Ethereum blockchain. Since Ponzi schemes need to return Ether to past investors, the information of all investors is stored in *storage*.

Opcodes of smart contracts. For smart contracts to work, they have to be compiled from their high-level languages (e.g., Solidity) into opcodes (also called operation codes) that can be executed by the EVM [29]. According to prior research [7] and our own observations, Ponzi smart contracts use four critical opcode instructions: *CALLER*, *CALL*, *SSTORE*, and *SLOAD*. For a Ponzi smart contract, *CALLER* adds the address of the account that used a smart contract to the *stack*, and is used to retrieve the new investor's account address. *SSTORE* copies data from the *stack* to *storage* and is used to permanently store the information of new investors. *SLOAD* reads a value from the *storage*, and retrieves information of previous investors in a Ponzi smart contract. *CALL* is used to transfer Ether to an address. Ponzi smart contracts use the instruction *CALL* to transfer Ether to the previous investor account addresses obtained using *SLOAD*.

Basic blocks, execution paths and control flow graph. Opcodes can be separated into groups of basic opcode blocks, where each one affects the *stack* in the same way and ends with either a condition leading to another basic opcode block or a termination instruction. Depending on the design logic of a smart contract, there can be different execution sequences of these basic opcode blocks, which are called *execution paths* in this paper. Each execution path carries out a task specified in the smart contract. We use a Control Flow Graph (CFG) to represent all the possible execution paths.

Ponzi schemes at opcode level. According to SADPonzi [7], the smart contracts of Ponzi schemes involve two types of critical actions that can be captured through opcodes: *investing* and *rewarding*. For investing actions, an investor invokes a transaction of a smart contract, which requires saving the investor information to the *storage* for future payment of rewards. Thus, an execution path that leverages *SSTORE* to save the data recorded by *CALLER* to the *storage* is considered as investing. The *rewarding* actions of Ponzi schemes refer to paying the profits from each new investment to prior investors, which must use the instruction *CALL*. Also, the *storing* actions are critical for identifying Ponzi schemes, as it is necessary to check whether the location storing the investor information in an *investing* action is the same as the storage location recording the investor information in the subsequent rewarding action.

4 OUR METHOD

PonziLens consists of three modules (Fig. 2): *opcode pre-processing*, *interpreter* and *visualization*. In the opcode pre-processing module, we collect the opcodes of a specific smart contract from EtherScan[1], a Block Explorer for Ethereum, and input them into teEther [16] to identify all basic opcode blocks and create the corresponding CFG. The interpreter module uses these basic opcode blocks and CFG to identify the crucial investing and rewarding execution paths, and the *storage* slots used by these paths. The visualization module shows the possible investing and rewarding paths and their interactions with the *storage* slots, enabling investors easily identify the possible Ponzi schemes in an intuitive way. Lastly, an opcode list is

[1] https://etherscan.io/

also included for a detailed investigation of the opcodes of a smart contract.

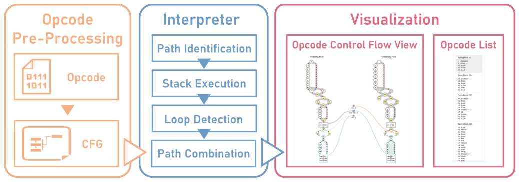


Figure 2: The architecture of *PonziLens* consists of three modules: opcode pre-processing, interpreter, and visualization.

4.1 Interpreter

The interpreter is designed to extract the critical information from the opcodes and CFG, and consists of four steps executed sequentially.

Path identification for a smart contract Path identification aims to collect all potential investing and rewarding paths from the CFG. We build all execution paths as Directed Acyclic Graphs (DAG), which are further categorised into investing paths that contain the opcode instructions *CALLER* and *SSTORE* and rewarding paths containing the opcode instructions *CALL*.

Stack execution Next, we need to determine if the investing and rewarding paths are properly executed by the EVM through the stack. In a Ponzi smart contract, an investing path should present a *CALLER* instruction on its *stack* when executing the instruction *SSTORE*, indicating that the investor's information is stored in the *Storage*. A rewarding flow should pick up an address obtained with the instruction *SLOAD* when transferring Ether to the previous investors using the instruction *CALL*. Through a process called stack execution, *teEther* [16] uses z3 [20] to run an execution path and identify what it does to the *stack*. We apply this method to check all the identified paths, and keep the ones that exhibit investing and rewarding behaviours on the *stack*. Stack execution also provides us with information on the *storage* slots involved in either the investing or rewarding paths, or both. Each slot is a number when used to store state variables, and SHA-256 hash is used to retrieve data structures such as an array or mappings on *storage*. This slot information will be used in the visualization module.

Loop detection. Ponzi schemes often use a rewarding execution path containing a loop starting with the instruction *CALL* to reward multiple investors. We modified the method used during path identification to determine which rewarding path exhibits this characteristic.

Path combination We combined investing and rewarding paths performed exactly the same way during stack execution into aggregated paths with a set of edges and nodes from the original execution paths to maximize the efficiency of our analysis.

4.2 Visualization

The visualization module helps investors to explore the features of a smart contract interactively and validate whether it is a Ponzi scheme. Specifically, an Opcode Control Flow View (Fig. 1A) shows the potential execution paths of a smart contract when an investor invokes an investment transaction using a smart contract. An opcode List (Fig. 1B) is incorporated to show the details of each basic opcode block in the CFG, helping investors further verify the insights from the Opcode Control Flow View.

Ponzi schemes have distinctive characteristics in their opcodes (Sec.3), which has also guided our visualization design. Specifically, we have considered the following two major opcode characteristics of Ponzi schemes (Fig. 1C):

C1. Investing flow and rewarding flow share one common execution path operating on the same storage slot. For a Ponzi scheme, the target address of the Ether transferring in the rewarding flow is obtained from the same *storage* slot where the prior investors' information is stored during the investing flow, as the invested Ether by a new investor will be paid as rewarding to prior investors. Also, such a payment is directly done when a new investment transaction is provoked, making the investing flow and rewarding flow share one common execution path.

C2. A loop path with a *CALL* instruction in the rewarding flow. The rewarding flow of the smart contract of Ponzi schemes often needs to pay Ether to multiple prior investors, which results in a loop in the execution path. Since *CALL* is necessary for transferring Ether in opcode of Ethereum, the loop also contains *CALL*.

4.2.1 Opcode Control Flow View. The Opcode Control Flow View (Fig. 1A) is designed to show the potential investing and rewarding paths and their interactions with the *storage* in EVM. The Ponzi Detection View consists of three parts: investing control flow (abbreviated as investing flow in Fig. 1a), rewarding control flow (abbreviated as rewarding flow in Fig. 1b), and storage interactions (Fig. 1c).

Investing flow and rewarding flow Investing control flow provides an overview of all aggregated investing paths. As shown in Fig. 1a, the investing flow is a directed graph. The basic opcode blocks in the CFG are represented by circular nodes with a label indicating its block index, and the execution path order is indicated by a black curve with arrows. As there can be overlaps between aggregated paths from the interpreter, we use a different color to wrap around each path's associated basic opcode blocks to represent it. This way, the investor can quickly identify the basic blocks that each path passes through. Similarly, the rewarding control flow shows all aggregated rewarding paths, which uses the same visual encodings as the investing flow. The investing flow and rewarding flow in one execution path are encoded in the same color to show that they will be executed in the same contract call. Given that only code loops with the *CALL* instruction are important, we visually encode them with the same color used to encode the rewarding paths they originate from. To help the investor visually differentiate between the directed-graph and the code loops, we also fill in the area that the loop path encloses with the same color. Furthermore, the basic blocks containing the critical opcodes like CALLER, CALL, and SSTORE, are indicated with a yellow background (Fig. 1A and Fig. 1B) can be unfolded by clicking to show the critical opcodes in it (Fig. 1d). The investors are also allowed to highlight one of the paths in the Opcode Control Flow View by hovering above the path.

Storage interactions The storage interactions are designed to display how the investing flow and rewarding flows interact with the *storage* in EVM and help investors verify whether the new investment is directly transferred to previous investors. We draw all *storage* slots accessed by the investing and rewarding flows, as shown in Fig. 1c. The type of data in each *storage* slot is encoded by different glyphs. In the Opcode Control View, the circle is used to visually represent a state variable, and the rectangle is used to represent an array. (Fig. 1g). A line is used to intuitively represent interactions between a path and one or more *storage* slots (Fig. 1c). When an investing path interacts with a *storage* slot, this visualizes the storing of investor information to that *storage* slot. When a rewarding path interacts with a *storage* slot, this visualizes the retrieval of past investor information from that *storage* slot so that Ether can be transferred to them. Interactions that belong to the same execution path are encoded with the same color. By analyzing the storage interactions of the investing flow and rewarding flow, it is easy and intuitive to verify whether the contract transfers Ether to the previous investors.

4.2.2 Opcode List. To help investors further confirm the insights obtained from the Opcode Control Flow View, we also show the original opcodes list separated according to the basic opcode block index. All nodes unfolded by investors in the Opcode Control Flow View are highlighted in the Opcode List, and the Opcode List will scroll to the position of the last block clicked by the investor.

5 USAGE SCENARIO

We showcase two usage scenarios with both Ponzi and non-Ponzi smart contracts to demonstrate the effectiveness of *PonziLens*.

5.1 Scenario 1: A Ponzi Smart Contract

We use *PonziLens* to explore the smart contract of a confirmed Ponzi scheme[2] that has been used by prior studies [4, 7].

Fig. 1A provides an overview of the investing and rewarding flows of this smart contract, where some opcode blocks in these paths lead to the storage slots as shown in Fig. 1c. We can see that the paths are encoded in three colors (red, blue, and green), indicating that there are three aggregated paths (i.e., *Path0*, *Path1*, and *Path2*) in this smart contract. Figs. 1a and 1b show that all the three paths are involved in both investing and rewarding flows, while Fig. 1c shows that *Path1* and *Path2*, indicated by the blue and green colors, congruently link their investing and rewarding flows through the same *storage* slot *185...94*. The insights here are two-fold. First, both investing and rewarding will be triggered when a transaction of this smart contract is provoked. Secondly, the invested Ether by new investors is probably paid to prior investors as rewards, as *Path1* and *Path2*, involved in both the investing flow and rewarding flow, operated on the same *storage* slot *185...94*.

To clearly check *Path1*, we can hover above the blue path to highlight it, as shown in Fig. 1C. Also, we can see that there is an execution loop in the rewarding flow with the *CALL* instruction presented (Fig. 1e), indicating that the smart contract is recursively sending Ether to the addresses of multiple prior investors.

By comparing all the above observations with the two major characteristics of a Ponzi scheme (Sec. 4.2), it is safe to confirm that this smart contract is a Ponzi scheme.

5.2 Scenario 2: A Smart Contract for Charity

We further use *PonziLens* to explore a smart contract for charity, since they are not Ponzi schemes but also transfer Ether from a large number of donors to the address of a charity, Specifically, we explored is EthPledge[3][4], a decentralized smart contract on Ethereum that allows people to donate money to a charity.

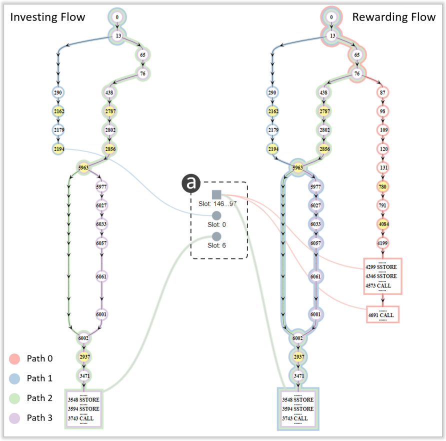


Figure 3: The opcode control flow of *EthPledge*, a smart contract for charity. *PonziLens* shows the investing flow and the rewarding flow, as well as their interactions with storage slots (a).

Fig. 3 shows the investing and rewarding flows of this smart contract. It demonstrates that this contract receives and transfers Ether, which is natural for a charity. However, there are no congruent execution paths from the investing flow to the rewarding flow linked by a location in *storage* (Fig. 3a). We can see that the execution paths involved in the investing and rewarding flows use different *storage* slots (Fig. 3a), indicating that investments cannot be transferred to prior investors. Given that the key characteristics of Ponzi schemes mentioned in Sec. 4.2 is not seen here, we can confidently conclude that this smart contract is NOT a Ponzi scheme.

Also, it is interesting to see that all the interactions between *storage* and the rewarding flow occur at the same *storage* slot *146...97* (the rectangle of Fig. 3a). It indicates that Ether is always transferred back to the addresses stored in an array of *storage*. This array probably records the information of all the charity organizations.

6 DISCUSSION AND CONCLUSION

In this paper, we propose a novel visual analytics system, *PonziLens*, for early identification of Ponzi smart contract. *PonziLens* can visualize all possible investing or rewarding execution paths in an aggregated manner as well as their interactions with the EVM storage, which reveals the critical features for identifying a Ponzi smart contract. We present two usage scenarios with two real smart contracts on Ethereum blockchain. The results indicate that *PonziLens* can help investors easily identify Ponzi schemes on Ethereum.

[2] Address: 0x0b230b071008bbb145b5bff27db01c9248f486b9

[3] Address: 0x10Ec03b714A2660581040c1A0329d88e381cA603

[4] https://www.ethpledge.com/

However, *PonziLens* is not without limitations. The current visual design may suffer from scalability issues when a smart contract has a huge number of basic opcode blocks and execution paths. In the future, we plan to improve the scalability of *PonziLens* by supporting the hierarchical aggregation of basic opcode blocks and execution paths. Further, it will be interesting to explore how our approach can be extended to the early detection of other smart contract frauds like honeypot contracts and pump-and-dump schemes. Besides, a quantitative user study is necessary future work to further evaluate the effectiveness and usability of *PonziLens*.

ACKNOWLEDGMENTS

This work was done during Xiaolin Wen's internship at the Singapore Management University (SMU) under the supervision of Dr. Yong Wang. This work was supported by the Singapore Ministry of Education (MOE) Academic Research Fund (AcRF) Tier 1 grant (Grant number: 20-C220-SMU-011), Lee Kong Chian Fellowship awarded to Dr. Yong Wang by SMU, and the National Natural Science Foundation of China (Grant number: 62172289). We would like to thank the anonymous reviewers for their feedback.

REFERENCES

[1] Mansoor Ahmed, Ilia Shumailov, and Ross Anderson. 2018. Tendrils of crime: Visualizing the diffusion of stolen bitcoins. In *Proceedings of International Workshop on Graphical Models for Security*. Springer, 1–12.

[2] Marc Artzrouni. 2009. The mathematics of Ponzi schemes. *Mathematical Social Sciences* 58, 2 (2009), 190–201.

[3] Thibault de Balthasar and Julio Hernandez-Castro. 2017. An analysis of bitcoin laundry services. In *Proceedings of Nordic Conference on Secure IT Systems*. Springer, 297–312.

[4] Massimo Bartoletti, Salvatore Carta, Tiziana Cimoli, and Roberto Saia. 2020. Dissecting Ponzi schemes on Ethereum: identification, analysis, and impact. *Future Generation Computer Systems* 102 (2020), 259–277.

[5] Massimo Bartoletti, Barbara Pes, and Sergio Serusi. 2018. Data mining for detecting bitcoin ponzi schemes. In *Proceedings of 2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*. IEEE, 75–84.

[6] Vitalik Buterin et al. 2014. A next-generation smart contract and decentralized application platform. *WHITE PAPER* 3, 37 (2014).

[7] Weimin Chen, Xinran Li, Yuting Sui, Ningyu He, Haoyu Wang, Lei Wu, and Xiapu Luo. 2021. Sadponzi: Detecting and characterizing ponzi schemes in ethereum smart contracts. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 5, 2 (2021), 1–30.

[8] Weili Chen, Zibin Zheng, Edith C-H Ngai, Peilin Zheng, and Yuren Zhou. 2019. Exploiting blockchain data to detect smart ponzi schemes on ethereum. *IEEE Access* 7 (2019), 37575–37586.

[9] Filippo Contro, Marco Crosara, Mariano Ceccato, and Mila Dalla Preda. 2021. EtherSolve: Computing an Accurate Control-Flow Graph from Ethereum Bytecode. In *Proceedings of 2021 IEEE/ACM 29th International Conference on Program Comprehension (ICPC)*. IEEE, 127–137.

[10] Chris Dannen. 2017. *Introducing Ethereum and solidity*. Vol. 1. Springer.

[11] Giuseppe Di Battista, Valentino Di Donato, Maurizio Patrignani, Maurizio Pizzonia, Vincenzo Roselli, and Roberto Tamassia. 2015. Bitconeview: visualization of flows in the bitcoin transaction graph. In *Proceedings of 2015 IEEE Symposium on Visualization for Cyber Security (VizSec)*. 1–8.

[12] Shuhui Fan, Shaojing Fu, Haoran Xu, and Xiaochun Cheng. 2021. Al-SPSD: Anti-leakage smart Ponzi schemes detection in blockchain. *Information Processing & Management* 58, 4 (2021), 102587.

[13] Xuezhi He, Tan Yang, and Liping Chen. 2022. CTRF: Ethereum-Based Ponzi Contract Identification. *Security and Communication Networks* 2022 (2022).

[14] Huiwen Hu and Yuedong Xu. 2021. SCSGuard: Deep Scam Detection for Ethereum Smart Contracts. *arXiv preprint arXiv:2105.10426* (2021).

[15] Eunjin Jung, Marion Le Tilly, Ashish Gehani, and Yunjie Ge. 2019. Data mining-based ethereum fraud detection. In *Proceedings of 2019 IEEE International Conference on Blockchain (Blockchain)*. IEEE, 266–273.

[16] Johannes Krupp and Christian Rossow. 2018. teEther: Gnawing at Ethereum to Automatically Exploit Smart Contracts. In *Proceedings of 27th USENIX Security Symposium (USENIX Security 18)*. 1317–1333.

[17] Hiroki Kuzuno and Christian Karam. 2017. Blockchain explorer: An analytical process and investigation environment for bitcoin. In *Proceedings of 2017 APWG Symposium on Electronic Crime Research (eCrime)*. 9–16.

[18] Shlomi Linoy, Natalia Stakhanova, and Suprio Ray. 2021. De-anonymizing Ethereum blockchain smart contracts through code attribution. *International Journal of Network Management* 31, 1 (2021), e2130.

[19] 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*. Springer, 41–56.

[20] Leonardo de Moura and Nikolaj Bjørner. 2008. Z3: An efficient SMT solver. In *Proceedings of International conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 337–340.

[21] Frédérique Oggier, Silivanxay Phetsouvanh, and Anwitaman Datta. 2018. BiVA: Bitcoin network visualization & analysis. In *Proceedings of 2018 IEEE international conference on Data Mining Workshops*. IEEE, 1469–1474.

[22] Giuseppe Antonio Pierro. 2021. Smart-graph: Graphical representations for smart contract on the ethereum blockchain. In *2021 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 708–714.

[23] Weisong Sun, Guangyao Xu, Zijiang Yang, and Zhenyu Chen. 2020. Early detection of smart ponzi scheme contracts based on behavior forest similarity. In *Proceedings of 2020 IEEE 20th International Conference on Software Quality, Reliability and Security*. IEEE, 297–309.

[24] Yujing Sun, Hao Xiong, Siu Ming Yiu, and Kwok Yan Lam. 2019. Bitvis: An interactive visualization system for bitcoin accounts analysis. In *Proceedings of 2019 Crypto Valley conference on blockchain technology*. IEEE, 21–25.

[25] Nick Szabo. 1996. Smart contracts: building blocks for digital markets. *EXTROPY: The Journal of Transhumanist Thought,(16)* 18, 2 (1996), 28.

[26] Sean Tan, Sourav S Bhowmick, Huey Eng Chua, and Xiaokui Xiao. 2020. LATTE: Visual construction of smart contracts. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 2713–2716.

[27] Kentaroh Toyoda, P Takis Mathiopoulos, and Tomoaki Ohtsuki. 2019. A novel methodology for hyip operators' bitcoin addresses identification. *IEEE Access* 7 (2019), 74835–74848.

[28] Kentaroh Toyoda, Tomoaki Ohtsuki, and P Takis Mathiopoulos. 2017. Identification of high yielding investment programs in bitcoin via transactions pattern analysis. In *Proceedings of GLOBECOM 2017-2017 IEEE Global Communications Conference*. IEEE, 1–6.

[29] Gavin Wood et al. 2014. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper* 151, 2014 (2014), 1–32.

[30] Wen Xiaolin, Wang Yong, Yue Xuanwu, Zhu Feida, and Zhu Min. 2023. NFTDisk: Visual Detection of Wash Trading in NFT Markets. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23)*.

[31] Yanmei Zhang, Siqian Kang, Wei Dai, Shiping Chen, and Jianming Zhu. 2021. Code Will Speak: Early detection of Ponzi Smart Contracts on Ethereum. In *Proceedings of 2021 IEEE International Conference on Services Computing*. IEEE, 301–308.