Ponzi Schemes on Blockchain [*]

Yifei Shuang[†] Ke Tang[‡] Yang You[§] Xi Zhao[¶]

June 18, 2023

Abstract

This paper analyzes 512 "vanilla" Ponzi scheme contracts on Ethereum by examining Solidity code with a machine learning algorithm and further manual validation. The unpunished Ponzi crime is quite sizable, ranging from 7,513 to 12,563 months under the current legal system. We investigate the Ponzi's features and incentive design and show that Ponzi schemes lure more victims and investment if (1) the interest rate is reasonably low, (2) the contract design is more innovative and sophisticated, (3) the Ponzi schemer exploits more affinity network, and early investors are more experienced with blockchain, and (4) incentive design encourages reinvestment, allow users to profit through their network, and charges lower fees.

JEL Classification: D26, D82, E60, G23, G28.
Keywords: Ponzi Scheme, Blockchain, Contract Design, Cyber-crime

[*]First draft: June 10th, 2023. We thank comments from Shiyang Huang, Chen Lin, Tse-Chun Lin, Xuewen Liu, Yang Liu, Fangzhou Lu, and Igor Makarov. All errors are our own. Xi Zhao acknowledges the funding support by the National Natural Science Foundation of China (Grant No. 72231007) and The Key Research and Development Projects of Shaanxi Province (Grant No. 2020ZDLGY09-08). Yang You, thanks support of the Shenzhen Fintech Grant (Grant No. SZRI2023-CRF-03).

[†]Xi'an Jiaotong University. Email: shuangyifei@stu.xjtu.edu.cn.

[‡]Institute of Economics, School of Social Science, Tsinghua University. Email: ketang@mail.tsinghua.edu.cn.

[§]University of Hong Kong. Email: yangyou@hku.hk.

[¶]Xi'an Jiaotong University. Email: zhaoxi1@mail.xjtu.edu.cn.

1. Introduction

The Ponzi scheme, a form of fraud paying earlier investors with money from recent investors, has a long history since the 1920s, named after Italian businessman Charles Ponzi. However, empirical analysis of the Ponzi scheme design is quite limited for two reasons: first, only large-scale and damaging Ponzi schemes can be observed (selection bias); second, the design of the Ponzi game is hard to observe as details are not publicly disclosed. We identify 512 Ponzi game contracts on Ethereum to overcome these two difficulties by examining Solidity code with a machine learning algorithm. Our sample is not subject to selection bias, and smart contracts enable us to quantify the full details of a Ponzi scheme with more than 60 variables.

Ponzi owner (schemer) can engineer and deploy smart contracts to function as a Ponzi scheme on the blockchain. Blockchain Ponzi games are different in the following dimensions than the traditional offline Ponzi scheme. First, the schemer and participants in Smart Ponzi can be anonymous, and regulators cannot discover the real-world identity unless the schemer discloses it. Second, smart contracts are tamper-proof and automatically executed; thus, no cheating is possible in the contract execution. Investors can directly interact with the Smart Ponzi, and money would flow according to the rule pre-programmed with Solidity. Third, all transaction records and contract codes are open to the public on the blockchain. If investors are sophisticated enough, they can fully understand the nature of Ponzi and the state of the contract, for example, how much fund remains in the investment pool.

We use five necessary elements for a smart contract to be identified as a Ponzi schemes: investor can deposit and receive payment from the smart contracts, fee addresses to collect commissions, the function of interest calculation, function to withdraw interest, and the existence of an address that only distribute balance to fee addresses and investors. We train a machine learning algorithm for the Ponzi detection, and manually verify all contracts with a predicted probability higher than 50%. Through our Ponzi detection algorithm, we find 512 smart contracts function as "vanilla" Ponzi schemes — blockchain ensures no information asymmetry or cheating for sophisticated players.

The first observation is that blockchain transparency cannot prohibit financial crimes. 512 Ponzi schemes receive 781,988 Ether and engage participation from 51,634 unique addresses. We find significant participation in Ponzi schemes operating on the blockchain, even if many schemers explicitly leave comments in Solidity code and tell participants that this contract is a Ponzi game. But still, many participants bet on the Ponzi scheme and expect to collect interest from the participants who engage later. We collect data on prosecution cases of U.S. Ponzi schemes and estimate the relationship between investment size and sentence period from real-world Ponzi schemes. We apply this relationship to the 512 blockchain Ponzi schemes, and the unsentenced crime ranges from 7,513 to 12,563 depending on the model choice and assumptions of ETH prices.

To characterize features of a Ponzi scheme, we start with features described in SEC alerts about Ponzi schemes and document that Ponzi schemes gain more victims and lure more investment if (1) the interest rate is reasonably low to make the Ponzi scheme more sustainable, (2) contract design is more novel and sophisticated, e.g., more functions, longer bytecode, and fewer comments in Solidity code, (3) contract owners lure in addresses which previously have transactions with the owner, and also early investors are more experienced with blockchain and deposit more funding in the Ponzi.

Then, we further parse out more features of Ponzi contract design from the solidity code to better understand how these Ponzi schemes are executed. We use Lasso, Ridge, XGboost, random forest, and neural network models to characterize the optimal contract design for a Ponzi scheme. A well-designed Ponzi shall include incentives to encourage existing investors to reinvest, allow users to profit by referring new users, and charge lower fees.

Our study is related to the literature on the Ponzi scheme and financial crime. Ponzi scheme is classified as a white-collar crime(Gottschalk (2017);Reurink (2018);). Madoff is the most infamous person who creates the Ponzi scheme(Lewis (2013)). With the development of technology, the Ponzi scheme evolves to new versions. Some Ponzi schemes based on the websites are known as high-yield Investment Programs (HYIP)(Moore, Han, and Clayton

(2012)) because of their promise of extremely High returns. Ponzi schemes also exist in virtual game worlds(Adrian (2010)). In the world of cryptocurrency, lots of Ponzi schemes and their variations exist due to the lack of supervision. Without the qualification verification for sellers and access mechanism for investors, people can easily engage in Ponzi schemes. Some of them are deposit scams that use virtual currencies, such as the MMM fund(Boshmaf et al. (2020)), which became notorious in Russia and was restarted in 2011 with the help of digital currencies.

Factors influencing investors' participation decision of the Ponzi scheme have been explored in several studies. (e.g., Moore, Han, and Clayton (2012);Rantala (2019);Tennant (2011);Lewis (2012);Frankel (2012);(Krugman (2008))). As for the factors of inviters, the financial experience of the inviters plays an essential role in promoting the Ponzi scheme. The frauds of Bernard Madoff and Robert Allen Stanford illustrate this concept(Lewis (2012)). The informal network is also used to attract investors(Owens and Shores (2010)).

Studies also examined the effects of owner demography, like age, education, and income, on Ponzi investment(Rantala (2019)). The reputation of the creator also plays an important role in the Ponzi scheme expansion. Demography, tolerance of risk, and FOMO (fear of missing out) are known factors of Ponzi investors. Investors with little financial literacy(Singh and Misra (2022)) are more likely to participate in the Ponzi scheme. Using econometric evidence from Jamaica, Tennant (2011) explains why investor risk exposure to Ponzi schemes using the theory of investor gullibility and risk tolerance. Studies show that people with specific demography are more likely to invest in the Ponzi scheme (Amoah (2018); Deliema, Shadel, and Pak (2020); DeLiema, Li, and Mottola (2023); Raval (2021)).

In addition, trust between inviters and investors is the main factor for the success of the Ponzi scheme, and finds that trust plays an important role in persuading investors to participate in the investment(Gurun, Stoffman, and Yonker (2018); Nguyen et al. (2023); Lo and Kan (2023)). The Ponzi scheme's rule, guarantee, and marketing channel also play an important role in its promotion. Ponzi schemes use a complex rule to disguise themselves as investment projects with unique and secret opportunities (Frankel (2012)). The guarantees traditional Ponzi scheme owners give investors are from the fake financial report or tricky that give back money to some of the investors instead of all investors (Deason, Rajgopal, and Waymire (2015)). With these guarantees, Ponzi schemes can attract more participants and live longer since they have a low actual interest rate and withdrawal rate (Artzrouni (2009)). As for the marketing channel, previous cases show that Ponzi schemers use an affinity referal system (Rantala (2019)) or the internet to attract participants (Moore, Han, and Clayton (2012)).

The rest of this paper is organized as follows. Section 2 describes our detection procedure of Ponzi schemes and estimates the associated investment scale and unsentenced blockchain crime. Section 3 studies the relationship between the size of Ponzi schemes and well-known factors of financial crimes. Section 4 further investigates the Ponzi contract determinant and employs machine learning approaches to elicit "optimal" contract design. Section **??** provides a theoretical framework for the Ponzi scheme to unify our empirical findings. Section 5 concludes the paper.

2. Ponzi Schemes on Blockchain

According to the definition of the U.S. Securities and Exchange Commission (SEC), a Ponzi scheme is an investment scam that involves the payment of purported returns to existing investors from funds contributed by new investors (investor.gov (2022)).[1] Blockchain technology also yields more convenience for Ponzi scheme creation: First, any user can deploy Ponzi scheme contracts on blockchain; second, it is hard to trace issuer's identity through wallet addresses or smart contracts; thus there is no legal punishment towards ponzi schemes on blockchain, at least so far; third, the nature of blockchain does not require minimum qualification of smart contract issuers or impose any investor protection rule for average participants.

[1]Ponzi schemes are also known as high-yield Investment Programs (HYIP) as a Ponzi game typically allures investors with extremely high returns (Moore, Han, and Clayton (2012)).

2.1. Detection of Ponzi Schemes on Blockchain

2.1.1. Ponzi Detection Principles

Smart Ponzi schemes on Ethereum are written in Solidity code, and their source codes are mostly publicly available to anyone on blockchain. We follow Bartoletti et al. (2020) to identify Ponzi schemes on Ethereum by detecting four principles: First, the contract distributes money among investors. Second, the contract receives money only from the investors but not others. Third, each investor makes a profit only if enough investors invest enough money afterward. Last, the later an investor joins the contract, the greater risk of losing his investment. Based on these principles, we further detail criteria to identify smart Ponzi based on Solidity code and transaction patterns:

- C1 (The Definition of Investors): The Smart Ponzi must have functions to register the address of users who send money to the contract as investors. Moreover, investors are able to access the payable function of Smart Ponzi.

- C2 (The Definition of Fee Address): The Smart Ponzi may have functions or variables to register the owner's address to receive the development fee and other fees as fee addresses.

- C3 (The Function of Interest): The interest must be a monotone increasing function of invest time and invest amount.

- C4 (The Mechanism of Withdraw): The solidity code of Smart Ponzi must have functions that can deliver ETH to the existing investors or other fee addresses which can be queried from the solidity code. These functions can be executed by investors or scammers.

- C5 (The Destination of Balance): The balance of Smart Ponzi can only be delivered to the Investors or Fee Address. The payable function of the solidity code must restrict the address that received ETH from the smart contract.

These criteria identify the definition of investor and fee address, and describe the core mechanism of Smart Ponzi including interest calculation, withdraw and allocation of contract balance. To describe the transaction pattern of smart Ponzi identify by our five criteria, Figure IA1 presents a typical Ponzi scheme payment flow. The color of the point in Figure IA1 shows that earlier investors easily reach break-even. The function in raw 11-24 describes the mechanism to withdraw the interest(C4). Only the investors that invested can withdraw the interest in raw 13(C5). Each column refers to an investor; the point refers to a transaction he made. The color of point refers to whether the investor reaches break-even. The shape refers to whether the transaction is investing or withdrawn. Each column in Figure IA1 is started by the cross, meaning that investors must invest before withdrawing from this contract.

Figure IA2 gives an example of the Solidity Code of the Ponzi scheme. In raw 6-9 and function in raw 11-24, the contract defines the investor variable and the way to become the investor (C1). However, the contract does not have fee settings or fee addresses (C2). The interest is calculated in the raw 13-20, where the interest rate is 10% per day, a monotone increasing function of investment time and amount(C3).

We exclude some contracts from the Ponzi since they conflict with our criteria as shown in Table 1. Besides Eth, users can create their crypto-tokens called layer 2 Token on Ethereum (The Eth is layer 1 Token). The token is based on the smart contract, and the contract that represents the Token is called "Token contract". "Fomo3D" is a monentory game that similar to the Ponzi scheme. Figure IA3 gives an example of lottery project code. People can buy the key to engage the game or buy the P3D token to be the shareholder of game. The last person to buy a key at the end of a round wins the pot. Besides the pot, player can get the refer reward from the later players. It is contrast to the C3 since the last buyers can win the bot. Besides, the balance is delivered to the P3D token holders which is not the investors in C1. "Gambling contract" include lottery, dice and other bet markets. "Gaming contract" is the contract used for the online game using Ethereum as payment or database. Figure IA4 gives an example of lottery project code. It is contrast to the C3 since the revenue function is not a monotone increasing function of invest time and invest amount. "ICO contract" is used for Initial Coin Offering(ICO), which raise ETH or stable coin from other users and send them layer 2 token as a reward. Figure IA5 gives an example of ICO project code. It is in contrast to the C4 since the contract does not have the function to withdraw interests from investors.

2.1.2. Seed Contracts of Smart Ponzi

Bartoletti et al. (2020) has also constructed a database of Ponzi schemes on the blockchain. Using Bartoletti's dataset, some artificial intelligence methods are applied to detect Ponzi contracts(Chen et al. (2018)).

To detect the smart Ponzi schemes, we first collect original smart contracts labeled as Ponzi by searching various sources. We review the code of these smart contracts and establish a new dataset including 331 Ponzi schemes, and 133 pyramids[2], and 4659 others. We split the data set as follows: 75% training data and 25% test data.

2.1.3. Detection Algorithm

We adopt a machine learning algorithm to detect Ponzi schemes from open-source smart contracts on Ethereum. "Contract Creation Bytecode" is the Bytecode of smart contract when it was established. Smart contracts with similar functions may have similar bytecode especially when they are open-source. By using a NLP (Natural Language Processing) method called TF-IDF (Aizawa (2003)), we calculate a feature vector for each smart contract by the following step: Step 1, we compile the Bytecode to the Opcode, and split them in the specific code segment. Step 2, we calculate the Term frequency $\mathrm{TF}_{\mathrm{i,j}}$ of specific code segmentn_{i}

$$\mathrm{TF}_{\mathrm{i,j}} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

Where $\mathrm{n}_{\mathrm{i,j}}$ refers the occurrence times of code segment $\mathrm{n}_{\mathrm{i,j}}$ in contract d_{j}, and k is the total number of code segments included in contract d_{j}.

[2]Meanwhile, we also collect samples of pyramid schemes. Pyramid refers to a type of fraud in which participants profit almost exclusively through recruiting other people to participate in the program

Step 3, we calculate the Inverse document frequency of specific code segment $\mathrm{n_i}$.

$$\mathrm{IDF_i} = \log \frac{|D|}{|\{j : t_i \in d_j\}| + 1}$$

Where $|D|$ refers to the total number of sample contract, $\{j : t_i \in d_j\}$ is the total number of contracts $\mathrm{d_j}$ contains the code fragment $\mathrm{n_i}$. In the last step, we multiply the TF and IDF, and get the feature vector $\mathrm{TFIDF_j}$ of smart contract $\mathrm{d_j}$.

$$\mathrm{TFIDF_{i,j}} = \mathrm{TF_{i,j}} \times \mathrm{IDF_i}$$

$$\mathrm{TFIDF_j} = (\mathrm{TFIDF}_{1,j}, \mathrm{TFIDF}_{2,j} \cdots\cdots)^T$$

Besides, we calculated two variables that whether the contract meet two smart contract Standard: ERC20 and ERC721. The feature factor of of smart contract j is

$$\mathrm{Feature_j} = (\mathrm{TFIDF}_{1,j}, \mathrm{TFIDF}_{2,j}, \cdots\cdots, ERC20, ERC721)^T$$

We select the XGBoost as the basic model to predict the positive probability of sample. Since the data is imbalanced, so we adopted the Synthetic Minority Over-sampling Technique (SMOTE) algorithm to generate more positive samples to train the model (Chawla et al. (2002)). Figure IA6 reports the roc curve of our detection algorithm by using the test dataset. The AUC score is 0.95, which shows that our detection algorithm has a good performance.

Table IA1 reports the performance of our detection algorithm. We use there classification model to detect the Smart Ponzi based on the same code features. Raw data is divided as training and test dataset. To choose the most suitable model, we use three metrics to evaluate the performance based on the test dataset, including EER, AUC and F1 score based on the Precision and Recall. XGBoost algorithm superior to other models on the EER and F1 score metrics. Although XGBoost algorithm doesn't have the best performance on the Precision and Recall metrics, it is not the worst among three models, shows that XGBoost model have the balanced performance on the two dimensions of the F1 score.

In the end, we summarize the results from classifiers and manually validate the "true results” by reviewing the codes of suspected contracts. We use this machine learning algorithm to detect smart contracts created from July 30th, 2015 to August 8th, 2019. Both the codes of original smart Ponzi schemes and the new detected ones are inspected to confirm if they meet the four principles of smart Ponzi schemes (including pyramids). Figure IA7 reports the precision of our detection method for different probability predicted. With the probability of Ponzi given by detection algorithm increase, the precision of algorithm increases from 0.07 to 0.79.

2.2. Ponzi Performance on Blockchain

We start with characterize our Ponzi schemes on blockchain in our sample.[3] To quantify measure the “success” of a Ponzi scheme, our paper focuses on the following three dimensions: First, we measure the number of participants by wallet addresses (e.g. interchangeable with participants henceforth) which invest money in the Ponzi game. Second, we measure investment scale by the total number of Ethers transferred to the Ponzi scheme address excluding ETH contributed by the owner (who deployed the Ponzi smart contract), and Third, we calculate the life span defined from the deployment date to the end date, which is defined as the first date when there is no new investment in this smart contract in the further 30 days.

Figure 1 plots shares of new participants, new investment, and the survival rate. Panel A reports the full sample results, and Panel B reports the 48 “established” Ponzi schemes with more than 100 addresses involved and the survival period is more than 10 days. With time goes by, average share and alive contract percentage steady decline. Figure IA8 plots the distribution of our three core performance metrics: the number of participants, the total investment amount (in ETH), and the survival time (duration). Figure 2 shows the fat-tailed distribution of three performance measures by estimating Zipf’s law with the largest 50 Ponzi schemes.

$$log(Performance\_Rank_i) = log(Performance_i) + \gamma + \epsilon_i$$

[3]Henceforth, we use “smart Ponzi”, “Ponzi on blockchain”, “blockchain Ponzi”, “Ponzi on Ethereum” interchangeably.

The slope in Panel A is -0.799 (s.e.=0.016) and R-squared is 98.09%, the slope in Panel B is -1.098 (s.e.=0.016) and R-squared is 99.02%, and the slope in Panel C is -1.788 (s.e.=0.031) and R-squared is 98.62%.

2.3. Extend profit of Ponzi Scammers

In this section, we describe the extend profit of ponzi scammers from the Ponzi schemes on Ethereum. There are 537 scammer accounts including the creators' addresses(schemers' address) and addresses which receive fee. Among 512 Smart Ponzi, only 276 contracts have transactions with the scammer accounts. Figure 3 describe the distribution of profit of Scammers. More than half scammers make profit from the Ponzi schemes while most success scammer have 27617.08 ETH income and 24934.17 profit ETH. Figure 4 provides the relationship between the size of Ponzi scheme and scammers' Profit. There are strong correlation between the size and scammers' Profit with slope is 0.798 (s.e.=0.054) and R-squared is 61.07% in Panel A, and the slope in Panel B is 0.656 (s.e.=0.038) and R-squared is 80.33%. The slope in Panel C is 0.339 (s.e.=0.067) and R-squared is 11.31%.

To quantify the extended profit from the affinity, the results are shown in Table 2. Panel A use a dummy variable as dependent whether the user make profit from the contract participation while Panel B use the users' Profit as dependent variable. In panel A column(6), after control contract effect, fee account and creator account have extend 61.3% (s.e.= 0.023) probability to make profit from the Ponzi scheme. Their friends have extend 6.1% probability to make profit, but the effect is not significant statistically. Other users only have 32.4% probability to make profit. As for the profit value, fee and creator account have 105.346 ETH than crowd, while their friend have extend 22.523 ETH. Other investors loss 1.394 ETH on average. It is worth pointing out that the summarize of all users' profit is less than zero since some ETH is locked in the Ponzi Contract and cannot be withdrawn for various reasons such as contract restriction.

2.4. Ponzi Schemes in Real World

Ponzi schemes are named after Charles Ponzi, who duped investors in the 1920s with a postage stamp speculation scheme. The key feature of Ponzi schemes requires a stable flow of new money to survive. When it becomes hard to recruit new investors, or when large numbers of existing investors cash out, these schemes will collapse soon.

Figure IA9 reports the Ponzi schemes sentenced in the US from 2008 to 2020. The data is collected from a database[4] which collected the sentencing news of Ponzi from SEC and other media. We download the database and transform the data by dropping duplicate samples, samples miss the sentence news link and sentences miss the amount and samples sentence month.

Figure IA9 shows that the peak amount occurred in 2008, when Bernard Madoff's Ponzi Scheme raised about 17.4 billion USD, which is beyond the total amount in any other year from 2009 to 2020. The sentence number of the Ponzi scheme is decreasing in the last 10 years, the reason may be that the Ponzi schemes in recent years are not sentenced since the progress of the sentence need a long period. As for the sentenced Ponzi schemes, the average of the total amount from 2008 to 2020 is 3.96 billion and the median is 1.33 billion.

Figure IA10 estimates the total investment amount (in USD) and imprisonment period (sentence months) after litigation in real-world top 50, 100, and 300 Ponzi schemes. The sentence period (Panels A to C) and investment amount (Panels D to F) both follow a nicely fitted linear relationship between the value and ranking. The largest Madoff scandal was also entitled to the longest sentence period of 150 years.

Lastly, we plot binscatter to show the relationship between sentence length (months) and investment amount from real-world Ponzi schemes in Figure IA11. Table IA2 formally runs the log-log specifications regressing the sentence period in months on Ponzi scheme investment amount. We estimate the linear relationship with all 675 ponzis in Columns (1) and with the largest 300 ponzi in Columns (3). The slopes, 0.216 versus 0.235, are quite similar in both samples. In Columns (2) and (4), we impose the restriction that the constant term is zero

[4]https://www.ponzitracker.com/ponzi-database

as no fraudulent investment shall correspond to no punishment. The slopes are higher, 0.286 and 0.281, in the no-constant regressions.

2.5. Unsentenced Imprisonment of Blockchain Ponzi

In this section, we borrow the linear models estimated from the real-world Ponzi and estimate the unsentenced crime of Ponzi schemes on the blockchain. Table IA3 provides the estimates of unsentenced crimes based on different linear models and ETH price assumptions. In Panel A, under the assumption that ETH price equals 2,000 USD, the no-constant model implies 7,513 months of imprisonment period, and the with-constant model implies 10,752 months. In Panel B, under the assumption that ETH peak price equals 4,867.97 USD, the no-constant model implies 9,289 months of imprisonment period, and the with-constant model implies 12,563 months.

3. Determinants of Ponzi Schemes

We first quantify a Ponzi scheme's success with the following three measures: the number of participants[5], the total investment amount is defined as the total amount of Ether deposited into the Ponzi scheme. If Smart Ponzi has not received a new investment within 30 days of the last investment, we set that investment as the last successful investment and calculate the life span. The investment value is not zero and it is verifed on the Ethereum before 10950000blocks. Table 3 Panel A reports the summary statistics of these three measures. A Ponzi scheme interacts with 100.84 addresses on average, receives 1527.32 Ether in total, and is active for 37.86 days. However, all three distributions tend to be very skewed: the largest Ponzi scheme Fairwin gains 19,069 addresses, and 693,146.56 ETH. The most long-living Ponzi game ShareholderVomer survived for 303 active days from July 5,2019 to May 3,2020. However, most Ponzi games attract little attention: the median Ponzi scheme only engages four participants, 0.58 ETH total investment, and lasts for two days.

[5]On the blockchain, it is almost impossible to precisely know whether two addresses are controlled by the same individual. The term "participants" refers to the wallet addresses throughout the paper.

Then, we quantify the important determinant of a Ponzi scheme inspired by SEC Ponzi scheme alerts on 2013 July 1st[6]. SEC characterized Ponzi schemes with the following common red flags that can be detected with our algorithm.[7]: First, High and overly consistent investment returns with little or no risk; Second, Secretive and/or complex strategies and fee structures. Third, It comes through someone with a shared affinity.

We by measuring these common characters: interest rate, affinity and initial displacement, smart contract complexity, and "innovation" of functions called in the smart contracts.

3.1. Interest rate

The first "red flag" of a Ponzi game is the high returns that allure greedy investors to participate.[8] A higher interest rate is a double-edged sword: a higher promised interest rate lures more participants and makes the Ponzi scheme more attractive, on the other side, an outrageous interest rate can deplete the deposit faster and crash the Ponzi earlier. It is an empirical question to investigate the relationship between the interest rate and ponzi game size.

Figure 5 Panel A plots the coefficient of the low-interest rate in the uni-variate regression and the performance of the Ponzi scheme after several days. The coefficients are reported in Table IA4. In Panel A1, the relationship between low-interest rates and new participants is U-shape. For all participants, if the low interest rate decrease 1%, the participants may increase 0.08%(s.e. = 0.038). The slope of low-interest rate to participants decrease from 0.005(s.e.=0.034) on first day to the -0.040(s.e.=0.04) for 21 days, then increase to 0.084(s.e.=0.050) for 60 days.

Figure 5 Panel A2 plots the relationship between low-interest rate and incremental investment after X days. The slope of low-interest rate to all investment is 0.021 (s.e. = 0.084). The slope of low-interest rate to incremental investment decrease from ,

[6]https://www.sec.gov/investor/alerts/enforcement

[7]Several red flags do not apply as there are no specific regulations to release smart contracts, for example, (1) Unregistered investments, (2) Unlicensed sellers, (3)Issues with paperwork, (4) No minimum investor qualifications, and (5) Difficulty receiving payments.

[8]https://www.investor.gov/protect-your-investments/fraud/types-fraud/ponzi-scheme

Panel A3 refers to the relationship between high interest rate and new participants. It is U-shape like Panel A1. From second day to 30 days, the coefficient is negative while the coefficient is positive in the other days. For total participants, the 1% increase of the high interest rate may increase participants 0.084%(s.e. =0.041). Panel A4 refers to the relationship between incremental investment and high interest rate. For all incremental investment, if the low interest rate increase 1%, the incremental investment may increase 0.026%(s.e. =0.044). Panel A5 refers to the relationship between high interest rate and payment interval. It is ascending from 0 day to 30 days, the coefficient is positive from first day but not significant statically. For total participants, the 1 day increase of the payment interval may decrease participants 0.005%(s.e. =0.000). Panel A6 refers to the relationship between incremental investment and payment interval. Size decrease by 0.01% (s.e. =0.007) when payment interval increase 1 day.

3.2. Contract Complexity and Innovation

To examine the relationship between the contract features and performance, we conduct a cross-sectional regression analysis with Success, Number of Participants and Investment as dependent variables. The independent variables are Original_name, Original_fun, Length of Bytecode and Function Numbers.

To measure the originality of the contract name, we use a dummy variable called Original_name as a proxy. Original_name is a dummy variable that indicates whether the Smart Ponzi is original or copied from another Smart Ponzi based on contract name. The steps are as follows. First, we download the contract names from Etherscan.io. Then, we convert the contract names to lowercase and remove the stopwords in them (stopwords include numbers and punctuation). Third, we use the BOW (bag of words) model to encode each contract name into a sparse vector. Fourth, we use the Birch algorithm with a 0.5 threshold to calculate the similarity among different Smart Ponzis and cluster them accordingly. Finally, we obtain 215 clusters of Smart Ponzis. In each group, we label the earliest created Smart Ponzis as original (Original_name = 1), and label the rest as copycats (Original_name = 0). Specifically, if some

Smart Ponzis are created on the same day, that is the earliest day, we label them as original (Original_name = 1).

We further define a variable called Original_fun to measure the originality of the contract functions. This is a dummy variable that indicates whether the Smart Ponzi is original or copied from another Smart Ponzi based on contract functions. The steps for calculating Original_fun are as follows. First, we use the BOW (bag of words) model to encode the ABI (application binary interface, which describes the interface of functions and events of the contract) of each smart contract into a sparse vector. Then, we use the Birch algorithm to calculate the similarity among different Smart Ponzis and cluster them with a 0.5 threshold. Finally, we obtain 196 clusters of Smart Ponzis. In each cluster, we label the earliest created Smart Ponzis as original (Original_fun = 1) and the rest as copycats (Original_fun = 0). Specifically, if some Smart Ponzis are created on the same day as the earliest day, we label them as original (Original_fun = 1).

To test whether that originality can enhance the performance of Smart Ponzi, we use linear regression to examine the relationship between originality and performance. Originality is measured by clustering the solidity code or the contract names. We report the results in Table 4. In column (2), we find that the contract function's originality of increases the probability of success by 22.7% (s.e.=0.068) with the control of different contract clusters. The originality of function increases participants by 120.5% (s.e.=0.207), investment by 157.0% (s.e.=0.521), and life span by 53.7% (s.e.=0.290). Moreover, originality also helps scammers get more profit (slope= 1.431, s.e.=0.644).

However, the originality of contract function does not equal originality of contract name. We add originality of contract name as an independent variable to the regression in Table 5. In a univariate regression, we find that originality of contract name helps the success and performance of Smart Ponzi, as shown in Table 5 columns (1), (3), (5) and (7). However, the coefficient is not significant at the 10% level. If a contract is original in both name and function, its performance is worse than that of a contract only original in function, but better than that of a contract only original in name.In column (2), we find that originality of name decreases the probability of success by 1.2% (s.e.=0.025) and originality of function increases it by 11.1% (s.e.=0.027). Originality of function increases participants by 60.7% (s.e.=0.176) and investment by 84.6% (s.e.=0.209), while originality of name decreases them by 1.5% (s.e.=0.169) and 20.5% (s.e.=0.200), respectively. Regarding the dynamic effect, Panel B in Figure 5 shows the originality for the surviving investment and participants, with all variables in Figure 5 normalized. The contract originality has a positive dynamic effect, as shown in Panels B1 and B2, with the coefficients reported in Table IA4. The coefficient of contract originality for participants fluctuates in the first week, ranging from 0.320 (s.e.=0.091) on the first day to 0.333 (s.e.=0.092) on the day five. The coefficient of contract originality for participants shrinks from 0.370 (s.e.=0.092) for 10 days to 0.207 (s.e.=0.094) for 60 days.

Moreover, the success of the original Ponzi schemes may be related to the high probability of having children. We show the relationship between the probability of having children and parent performance in Figure 6, which characterizes the probability of a child contract as a function of parent contract performance identified by function. As the size of Ponzi schemes increases, the probability increases from less than 10% to almost 80%. We show additional analysis in Figure IA12, where we calculate the probability of having children based on the contract name. The results are consistent with Figure 6, where the probability of having children has a positive correlation with the performance of the Ponzi scheme.

As shown in Table 6, parent success is positively associated with child success. In Panel A, we calculate the average size of parents and children in each cluster to measure their performance. In Panel B, we measure the size of parents and children by selecting the most successful contract with the maximum size in each cluster. In Panel C, we measure the size of parents and children by calculating their cumulative size as the sum of parent or child size in each cluster. For the average size, child participants, investment, and life span can be predicted by parent size by 17.2%, 12.4% and 18.6%, respectively. For the most successful child with the maximum size among other children, participants, investment and life span can be predicted by parent size by 22.3%, 18.2% and 27.3%, respectively. Parent size also has a significant correlation with the sum size of their child, where correlation coefficients are

24.7%, 19.7% and 37.3% for participants, investment and life span, respectively.We use three proxies to measure parent group or child group performance in Table IA5.The results also show a positive correlation between parent and child size. Pearson's correlation coefficients are all positive and significant at the 5% level. In each contract cluster, if the parent is more successful, the parent share will be higher, as shown in Figure 7. This means that if the original is more successful, the copycats are less likely to achieve the same level of parent performance.

When deploying a smart contract in Ethereum, the creator should compile the source code and send a "Contract Creation Transaction" to a new address representing the contract. This transaction delivers a specific bytecode called "Contract Creation Bytecode" to the target address. EVM establishes a smart contract's "Runtime Bytecode" by excluding the constructor logic and parameters of the "Contract Creation Bytecode." The Runtime Bytecode is stored in the Ethereum Network and interacts with the contract. Bytecode length is the natural logarithm of Contract Creation Bytecode length, which measures Contract Complexity. Function numbers are another proxy for contract innovation, referring to the defined functions in the solidity code of a smart contract.

We report the results in Table 7. The Contract Complexity variables are positive in column (1), showing that complexity contributes to the success of the Ponzi Scheme, with a coefficient of 2.8% (s.e. = 0.014). The Contract Originality increases participants by about 23.6% (s.e. = 0.089) and investment by 18.6% (s.e. = 0.108) in columns (3) and (5). For another proxy of complexity, Length of Bytecode, the effect is also positive on success. 1% increase in Length of Bytecode increases participants by 22% (s.e.=0.079) and investment by 21.4% (s.e.=0.099). We find the same effect for Function Numbers: Smart Ponzi with more functions has more participants and investment. This result is consistent with the previous one, suggesting that complex Smart Ponzi has more participants and investment than simple contract, and is therefore robust to alternative proxy variables of contract complexity. For the dynamic performance of Smart Ponzi, we show the results in Figure 5 Panel B and report the coefficients in Table IA4. After centralization, the slope of contract originality for participants all time is 0.320 (s.e.=0.091), and the slope increases from 0.334 (s.e.=0.091) for the first day to 0.370 (s.e.=0.092) for ten days, then decreases to 0.207 (s.e.=0.094) for the first 60 days. For investment, the slope of contract originality shrinks since the first day. On the first day, the slope is 0.335 (s.e.=0.090); for 10 days it is 0.290 (s.e.=0.088), and for 21 days, it is 0.199 (s.e.=0.084). For 60 days, the slope increases to 0.206 (s.e.=0.084).

3.3. Affinity frauds and Initial Displacement

Affinity frauds[9] are commonly accused for fraudulent investment frauds. Financial frauds tend to spread through informal private networks and attract new participants to the scheme by the nature of financial crime.

We test whether Ponzi schemers who exploit their affinity network are more likely to succeed in Table 8. First, we measure schemer experience on Ethereum by the number of transactions, account life, and the number of accounts they interacted with before launching the Ponzi scheme (connected accounts). We find no evidence that experienced schemers outperform systematically. For Ponzi performance, we find that the slopes of the number of transactions (slope = -0.038, s.e.= 0.051), account life (slope = -0.038, s.e.= 0.043) and connected accounts (slope = -0.061, s.e.= 0.059) are negative and not significant for the number of participants. The slopes of number of transactions (slope = -0.040, s.e.= 0.066), account life (slope = -0.010, s.e.= 0.052) and connected accounts (slope = -0.085, s.e.= 0.067) are negative and not significant for total investment A 1% increase in schemer transactions increases Ponzi scheme life by 0.1% (s.e.= 0.037). A 1% increase in account life results in a 1% increase in Ponzi scheme life (s.e.= 0.034). Connected accounts have a negative effect on Ponzi life (slope = -0.061, s.e.= 0.098).In addition to Ponzi performance, we find no evidence that experienced schemers earn more income or profit by creating a Ponzi scheme.The slopes of number of transactions (slope = 0.061, s.e.= 0.090), account life (slope = 0.059, s.e.= 0.068) and connected accounts (slope = 0.009, s.e.= 0.076) are positive but not significant for their income. After considering scammer income and investment, the slopes of number of trans-

[9]U.S. Securities and Exchange Commission (SEC) provides detailed information about affinity frauds in ff

actions (slope = 0.066, s.e.= 0.110), account life (slope = 0.079, s.e.= 0.085) and connected accounts (slope = -0.005, s.e.= 0.091) are not significant for their profit. Then, we measure the affinity exploitation by counting the number of connected accounts that invest in their Ponzi schemes in Column (4), and the share of participating accounts out of total connected accounts in Columns (5) and (6). For Ponzi performance, a 1% increase in invested connected accounts results in a 20.9% increase in Ponzi participants (s.e.= 0.098), a 9.3% increase in investment (s.e.= 0.128), and a 20.4% increase in life span (s.e.= 0.090). The slopes of share of participating accounts are 6.721 (s.e.=2.994) for Ponzi participants, 7.239 (s.e.=4.504) for investment, and 3.639 (s.e.=1.549) for life span, while the slopes of total connected accounts are 14.863 (s.e.=4.695) for Ponzi participants, 13.762 (s.e.=5.010) for investment, and 12.491 (s.e.=4.066) for life span. For scammer income and profit, a 1% increase in invested connected accounts results in a 1.9% decrease in scammer income (s.e.= 0.195) and a 6.4% decrease in scammer profit (s.e.= 0.206). Share of participating accounts has a positive but not significant effect on scammer income (slope = 5.900, s.e.= 5.511) and scammer profit (slope = 5.672, s.e.= 5.913), while the slopes of share of total connected accounts are 5.672 for scammer income (s.e.= 5.913) and 7.678 for scammer profit (s.e.= 8.172). These affinity exploitation measures predict more participants, larger investment, and longer ponzi duration with robust statistical significance.No evidence supports that affinity helps scammers get more income or profit.

The network of Ponzi Scheme participants might also be an important factor in the growth of a Ponzi scheme. On the Ethereum network, all investors use anonymous addresses to make transactions and investments. We define an experienced user as an address with a longer account life and more transactions. Table 9 reports the results of the regression for experience and performance. We find that the experience and account life of early investors positively contribute to a larger scheme. For the incremental participants, a 1% increase in investors' account life results in 19.7% more participants overall (s.e.=0.040), 27.4% more participants after 3 days (s.e.=0.033), and 26.0% more participants after 5 days (s.e.=0.030). A 1% increase in investors' transactions results in 22.4% more participants overall (s.e.=0.042), 22.6% more participants after 3 days (s.e.=0.045), and 25.1% more participants after 5 days (s.e.=0.041).

The slopes of account life to investment are 0.186 for all-time investment (s.e.=0.047), 0.218 for investment after 3 days (s.e.=0.038), and 0.206 for investment after 5 days (s.e.=0.035), while the slopes of account transactions to investment are 0.195 for all-time investment (s.e.=0.051), 0.165 for investment after 3 days (s.e.=0.053), and 0.196 for investment after 5 days (s.e.=0.049). The slopes of account life to the life span of Ponzi schemes are 0.50 for the life span (s.e.=0.030), 0.246 for the life span after 3 days (s.e.=0.029), and 0.277 for the life span after 5 days (s.e.=0.027), while the slopes of account transactions to life span are 0.039 for all time (s.e.=0.033), 0.165 for life span after 3 days (s.e.=0.031), and 0.211 for life span after 5 days (s.e.=0.028). For the incremental scammers' profit, a 1% increase in investors' account life results in 8.8% more scammers' profit overall (s.e.=0.062), 17.0% more scammers' profit after 3 days (s.e.=0.072), and 16.7% more scammers' profit after 5 days (s.e.=0.071). A 1% increase in investors' transactions results in 10.2% more scammers' profit overall (s.e.=0.057), 19.8% more scammers' profit after 3 days (s.e.=0.078), and 19.1% more scammers' profit after 5 days (s.e.=0.075).

In Figure 5 Panel C, we find that initial displacement is a determinant of Ponzi performance. The results show that the slope of the initial displacement decreases from 0.247 (s.e. = 0.045) for the participants who joined on the first day to 0.154 (s.e. = 0.057) for those who joined within the first five days, and then increases to 0.177 (s.e. = 0.083) for those who joined within 60 days. A similar pattern is observed for the initial investment, with the slope decreasing from 0.245 (s.e. = 0.055) to 0.108 (s.e. = 0.077) and then increasing to 0.128 (s.e. = 0.124). Moreover, this study explores whether there is a momentum effect of participants and investment. When a user invests in a Smart Contract, they can view its transaction records in blockchain browsers such as Etherscan.io, which provide information on the balance and the number of participants of smart contracts. These data may reflect the activity level of smart contract users and influence their future participation decisions.

Furthermore, we use cross-sectional regression and plot the coefficient in Figure 8. Specifically, we use the participants and investment as proxies of Ponzi performance and the initial participants and investment as proxies of initial performance. The coefficient on the previous participants indicates the effect of the number of users on future participation. A positive coefficient suggests that more users attract more participation, while a negative coefficient implies that more users reduce profitability and increase the abandonment of participation. A zero coefficient means that users do not pay attention to the number of previous investors. However, Figure 8 shows that the initial participants and investment are positively related to the incremental performance, as the line is above the x-axis. The effect of initial displacement diminishes over time, as the coefficient of the first day decreases. Moreover, this study finds that the first two days are better predictors of performance than the first five days, as they have the same trend but require less data collection.

4. Contract Design

With the full Solidity code, we can manually code up the features that describe the Ponzi mechanism, including interest settings, revenue sources, restrictions, contract comments, and fee structure. Based on these features, we use the horse racing method to find the important feature to predict the size of the Smart Ponzi. Then, we are trying to quantify the importance of different contract designs in determining the Ponzi scheme size.

4.1. Feature Construction

In this section, we construct some features to measure the mechanism of the Smart Ponzi. The summary statistics are shown in Table IA6.

4.1.1. Interest Settings

This subsection describes the interest variables in Smart Ponzi. The interest is calculated by the interest and hold times. The interest rate refers to the daily interest rate of investors' investments. Ethereum may generate about 6000 blocks per day, depending on the hash rate and difficulty of Ethereum minting. The contract may use different methods to measure time: 1. Set a fixed block number as the measure of time. For example, the interest is calculated by multiplying the number of past blocks and the interest rate. 2. Use UTC time as the measure of time. For example, the interest is calculated by multiplying the number of days past and the interest rate. Different time measures may make minimal variations to the interest, but we think this difference can be ignored since the fluctuation of block generation time is low.

The interest rate can be fixed or floating, where the latter means that the interest rate depends on other factors, such as the contract's current balance or the investment. $Rate_{rate}$ is a dummy variable for whether the interest rate is fixed. $Rate_{balance}$ is a dummy variable for whether Ponzi's interest rate depends on the contract's balance. $Rate_{investment}$ is a dummy variable for whether Ponzi's interest rate depends on the investors' investment.$Rate_{holdtime}$ is a dummy variable for whether the interest rate of Ponzi depends on the time after the investors' last withdrawal of investment.$Rate_{participants}$ is a dummy variable for whether Ponzi's interest rate depends on the contract's current investors.

4.1.2. Sources of Revenue

Besides the interest, Smart Ponzi investors may have other revenue sources from Smart Ponzi. This subsection describes other sources of revenue. $Random\ Reward(dummy)$ is a dummy variable indicating whether the Smart Ponzi has the mechanism to send rewards to the existing investors randomly, like lotteries. $Ref\ Ratio$ refers to the percentage of the investment of new investors that are paid to their referrers. Besides the direct referrers who refer the investors, some Smart Ponzi may pay indirect referrers who refer the referrers of new investors. In this variable, we only calculated the ratio that only pays direct referrers. $Rebate\,ratio$ refers to the percentage of investment of new investors that is paid back to them.

4.1.3. Restrictions

In this subsection, we construct variables to describe the restriction of investors' investments or withdrawals. $Manpay$ is a dummy variable that indicates whether Smart Ponzi will not automatically pay investors unless they send transactions to withdraw their interests. $Autopay$ is a dummy variable that indicates whether the Smart Ponzi pays old investors automatically so that they don't need to send transactions to withdraw their interests without the permission of the developer or manager of Smart Ponzi. *ProfitLimit* is a dummy variable set to 1 if an address cannot get more money when its rate of return has reached the limit specified in the Smart Ponzi scheme. *Withdrawals Interval* refers to the minutes that investors have to wait before they can withdraw next time, although the Ponzi contract has enough money to distribute among investors. *Exit* is a dummy variable that indicates whether Smart Ponzi gives investors a choice to exit the scheme so that they give up the opportunity of getting income anymore and get part of their investments. *Part Balance* is a dummy variable that indicates whether investors can get part of their money when the balance of the Smart Ponzi is below the amount proposed to pay investors. *Reinvest* is a dummy variable that indicates whether Smart Ponzi can reinvest in the same scheme when they reach the profit limit or are deleted from the payment queue. *FixedRevenue* is a dummy variable that indicates whether the payment from Smart Ponzi to the investors is a fixed amount. *Withdrawal Limit* is a dummy variable equal to 1 if investors are allowed to withdraw a limited number of times, such as once or twice.

4.1.4. Contract Comment

In this subsection, we construct two variables to describe the comment of the Smart Ponzi contract. *Comment* is a dummy variable if the source code verified by Etherscan.io has a code comment to explain its purpose or display the contact information of schemers. *Security Promise* is a dummy variable that equals one if the schemers promise security for this contract in their code comments. The promise uses sentences like "The contract has been tested for vulnerabilities!".

4.1.5. Fee Structure

In this subsection, we construct several variables to measure the fee of Smart Ponzi investment. These variables are derived from the Solidity code of the Smart Ponzi contract. *ExpenseRatio* is the percentage of the investment deducted for other expenses. To describe the expense ratio precisely, we divide the expense into four categories: advertisement fee ($AdvFee$), the public fee ($Public\,Fee$), development fee ($Dev\,Fee$), and other fees ($Other\,Fee$). Each fee is measured by the percentage of the investment. The use of fee is based on the statement by the schemer in the source code. $Adv\,Fee$ is the fee that is allocated to the advertisement. $Public\,Fee$ is the fee allocated to the common fund or other charity fund. $Dev\,Fee$ is the fee allocated to the schemer to develop the smart contract.

4.2. Predicting Ponzi with Contract Features

In this section, we aim to identify the most influential variables on the size of the Smart Ponzi among all the determinants. The analysis includes variables related to Interest Settings, Sources of Revenue, Restrictions, Contract Comment, and Fee Structure. We use OLS and LASSO regressions to select variables in Table 10. To address the potential heteroscedasticity in the regression, we transform some variables. The dependent variables are in logarithmic forms. The independent variables are transformed using two methods: converting continuous variables to dummy variables and converting continuous variables to rank form among all samples in ascending order. The dummy variables are labeled "(d)" while rank variables are labeled "(r)" in Table 10. All variables are normalized. Columns (1)-(4) report the OLS regression results, while Columns (5)-(8) report the LASSO regression results. The table shows only the significant variables where we omitted non-significant variables (P-value¿0.1) in columns (1)-(4) and variables with coefficients equal to zero in columns (5)-(8). The λ parameter used in LASSO regression is selected by the AIC criteria.

In Table 10, we identify some significant variables among all the determinants related to the source of revenue, interest settings, restrictions, contract comment, and fee structures. The results vary between the OLS regression and LASSO regression.

Regarding the interest settings, we find that one variables can predict the scammers' income. By using LASSO, we find three variables that can be used to predict both the performance of the Ponzi and the scammers' income. If the interest rate depends on the participants' holdtime, it will increase the investment from the investors (excluding the schemers' investment) by 7.1%. If the interest rate depends on the participants' investment, it will increase the investment from the investors (excluding the schemers' investment) by 5.7%, and increase the scammers' income by 13.3%, while using OLS regression yields an increase of 12.8% (s.e.=0.076) in the scammers' income. If Smart Ponzi adopts a fixed interest rate, its life span will shorten by 2.8%.

All the variables related to other sources of revenue are significant, including the *Ref Ratio*, *Rebate Ratio*, and *Random Reward*. *Ref Ratio* and *Rebate Ratio* are two variables that are in the rank form. The results show that momentary incentive in word-of-mouth marketing (*Ref Ratio*) helps more participants invest in the Smart Ponzi. The momentary incentive also attracts 9.5% more participants to Smart Ponzi in column (5). The results in column (2) and column (6) show that referral fee (*Ref Ratio*) may increase 13.9% (s.e.=0.066, by OLS estimation) or 13.1% (by LASSO estimation) investment on Smart Ponzi. *Ref Ratio* also increases 1.9% of scammers' income by LASSO in column (8). However, the coefficients of the *Rebate Ratio* in columns (2) and (4) suggest that the rebate may not attract investment as the schemers proposed. On the contrary, since the money needed to pay old investors is more than they invested, participants excluding schemers may invest 9.6% (s.e.=0.057, by OLS estimation) or 7.9% (by LASSO estimation) less on the Smart Ponzi with high *Rebate Ratio*. *Rebate Ratio* decreases 9.9% (0.059) of scammers' income by OLS. *Random Reward* is selected by the LASSO model when predicting the life span in column (7), which shows that the life span will increase by 3.6% if *Random Reward* was set in the Smart Ponzi. In addition, we perform a falsification test to check if the coefficients of all revenue variables are zero and a seemingly unrelated regression (SUR) to assess the correlation of the error terms of the *Ref Ratio* and *Rebate Ratio* across the equations. Table IA7 reports the joint test results and SUR coefficients of source revenue. We find that these revenue variables pass the SUR test but do not pass the joint test.

The explanatory variables that describe the restrictions are *Profit Limit*, *Reinvest*, *Fixed Revenue*, *Autopay*, *Withdraw Interval*, *Can Exit*, and *Part Balance*. If investors can reinvest in the Smart Ponzi, they may earn more interest before Smart Ponzi collapses.

Reinvest is significant in the OLS regression but excluded in the LASSO regression when predicting the *participants* (slope = 0.036, s.e. = 0.013) and *investment excluding schemers* (slope = 0.026, s.e. = 0.012) in Table 10. *Autopay* has a positive impact on the *life span* of Smart Ponzi, with a slope of 0.069 (s.e. = 0.041) based on OLS regression and a slope of 0.039 based on LASSO regression. *Fixed Revenue* limits the whale's investment in the contract and slows down the bank run process when the balance is low because investors cannot be paid up in one transaction. It increases the investment of Smart Ponzi by 3.8% (s.e. = 0.020) in OLS estimation. We find two factors that can decrease scammers' income: *Reinvest* increases it by 2.8% (s.e. = 0.001) according to the OLS model, and *Withdraw Interval* increases it by 7.5% (s.e. = 0.004) according to the LASSO model.

The effect of the *securitypromise* is negative, which contradicts the schemers' expectations. The promise sentences written in the source code are not verified by any security institute, so investors may not trust their promises. On the contrary, the promise may make investors doubt whether the contract is well-designed or copied from an existing contract. In LASSO regression, it may decrease the participants by 8.4% (9.4% by OLS model, s.e. = 0.052), the investment by 5.5%. Table IA8 reports the joint test of two variables and SUR regression results. Results show that *security promise* can be used to predict the performance of Ponzi. Moreover, we analyze how the keywords of comments may affect the performance of Ponzi. Table IA10 describes the effect of keywords in the Ponzi contract solidity code. Columns (1) and (2) report the joint test of different keyword variables in predicting the size of the Ponzi scheme. The joint test is not significant for participants but significant for investment amount with p-value = 0.011.

Four fee variables are selected by OLS and LASSO models. The selected fee variables are public fee (*Public Fee*), advertisement fee (*Adv Fee*), development fee (*Dev Fee*), and other fees (*Other Fee*), which may be related to the activities of advertisement, development, and public funds. Except for *Adv Fee*, other variables affect the performance. The LASSO regression shows that *Adv Fee* increases the life span by 5.5% in column (7). *Public Fee* hurts the participants (slope = -0.079, *s.e.* = 0.024), investment (slope = -0.077, s.e. = 0.022), and life span (slope = -0.035, *s.e.* = 0.012), while *Other Fee* has a negative effect on the participants (slope = -0.063, s.e. = 0.033), investment (slope = -0.063, s.e. = 0.024) and life span (slope = -0.035, s.e. = 0.017). Table IA9 reports the joint test results and sur coefficients of fee variables; these fee variables do not pass the SUR test and joint test.

4.3. The Importance of Contract Variables: Lasso Path

Lasso Path is a method which perform the interpretability of the LASSO model and the variable selection(Pedregosa et al. (2011)). The LASSO regression minimizes the following objective function:

$$\min_{w} \frac{1}{2n_{\text{samples}}} \|Xw - y\|_2^2 + \alpha\|w\|_1 \tag{1}$$

where the α is a constant and $\|w\|_1$ is the l1-norm of the coefficient vector. The value of α controls the degree of penalty. By selecting different values of α, we obtain different coefficients and plot them as a sequence called the LASSO path. The lasso Path of the contract design variables are plotted in Figure 9. The horizontal axis represents the values of $-log(\alpha)$, and the vertical axis represents the coefficient values. Each line represents a different variable. We measure the size of Smart Ponzi using three proxies, which we plot in different panels. In Panel A, we find that Ref Ratio (the olive line) is the most important variable, as it is the first variable selected when $-log(\alpha)$ is 0.93. Public Fee (the misty rose line) is another variable that has $-log(\alpha)$ less than 1. When $-log(\alpha)$ is set to 1.5, nine variables survive: five are positive (Ref Ratio, Adv Fee, Autopay, Reinvest and $Rate_{holdtime}$) and four are negative (Public Fee, Security Promise, Other Fee and Profit Limit).

We use the investment amount as a proxy for the size of Smart Ponzi in panel B. Ref Ratio (the olive line) is the most important variable, consistent with panel A. When $-log(\alpha)$ is set to 1.0, three variables survive: Ref Ratio, $Rate_{holdtime}$ and $Rate_{investment}$. However, $Rate_{investment}$, which is not important in panel A, is the third variable selected by the lasso as α decreases. This suggests that Smart Ponzi has an incentive mechanism to encourage investors to invest more money and signal their belief in the scheme, thereby attracting more investment. When $-log(\alpha)$ is set to 1.5, ten variables survive: six are positive (*Ref Ratio*,

$Rate_{holdtime}$, $Rate_{investment}$, $Withdraw\,Interval$, $Reinvest$ and $Autopay$) and four are negative ($Public\,Fee$, $Security\,Promise$, $Comment$ and $Rebate\,Ratio$).

In panel C, we use the investment excluding the schemers' investment as the dependent variable. Compared with the surviving variables when $-log(\alpha)$ equals 1.5 in panel B, $Withdraw\ Interval$ has $-log(\alpha)$ of 1.95, has 2.21 and $Adv\ Fee$ has 1.44. This means that excluding the schemers' investment, the $adv\ fee$ is quite important for attracting investment, whereas $Withdraw\ Interval$ and Comment are no longer important for affecting the size of Smart Ponzi.

4.4. The Importance of Contract Variables: Recursive feature elimination

In addition, we use six models to compare the importance of different variables in Figure 10. These models can be divided into two categories: linear models and tree models. Linear models include OLS, LASSO and Ridge regression, while tree models include decision tree, random forest and XGBoost. We apply the Recursive feature elimination (RFE) method to compare variable importance across different models (Guyon et al. (2002)). RFE eliminates variables recursively by estimating the R-squared of each model. The rank of variables in the figure refers to the order in which each variable is eliminated. The depth of color indicates the importance of variables by ordering them according to the absolute value of their coefficients. We use three proxies: participants, investment amount excluding schemers' investment and life span. The variable in dark blue is the most important one. Consistent with the previous results, Ref Ratio is the most explanatory variable in panel (a) and panel (b).

Apart from Ref Ratio, variables that describe the fee structure, such as Expense Ratio and other detailed fee variables, are influential. The results of linear models differ from those of tree models. Regarding the Fee settings, linear models favor detailed fees such as Adv Fee, Public Fee and Other Fee, whereas tree models favor the total fee ratio: Expense Ratio. In addition, linear models show that Profit Limit is an important variable related to restriction for predicting future participants. The tree models show that the important restriction variables are Withdraw Interval, part balance and Profit limit. Regarding the contract comment variables, Security Promise has a high correlation with Comment; linear models favor the former and tree models favor the latter. In panel C, for predicting life span, variables related to restriction are important but ignored in panels A and B. All restriction variables except Part Balance are more important than the average level in linear models.

5. Conclusion

With a history of over a century, Ponzi schemes have become active and mingled with new technology in the cyber-world, where regulation is lagged, and fraud detection technology is underdeveloped. This paper exploits the advantage of blockchain transparency, estimates the size of on-chain Ponzi schemes, and empirically evaluates Ponzi schemes' contract design. Our empirical evidence shows that the anti-fraud wisdom developed offline primarily still applies to combat the blockchain ponzi.

Our paper highlights the importance of parsing the smart contracts deployed on the blockchain. More governance and regulatory endeavor need to be made on blockchain to reduce cybercrime and alert risks of decentralized smart contracts. It is worth further investigating why participants engage in a Ponzi scheme: participants might need help understanding the risks of engaging in a contract program, particularly when the contract owner does not explicitly comment on its Solidity code. Or, participants firmly believe that they are not the last investor and can profit from new investments.

References

Adrian, Angela. 2010. “Beyond griefing: Virtual crime.” *Computer law & Security review* 26, 640–648.

Aizawa, Akiko. 2003. “An information-theoretic perspective of tf–idf measures.” *Information Processing & Management* 39, 45–65.

Amoah, Benjamin. 2018. “Mr Ponzi with fraud scheme is knocking: investors who may open.” *Global Business Review* 19, 1115–1128.

Artzrouni, Marc. 2009. “The mathematics of Ponzi schemes.” *Mathematical Social Sciences* 58, 190–201.

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

Boshmaf, Yazan, Charitha Elvitigala, Husam Al Jawaheri, Primal Wijesekera, and Mashael Al Sabah. 2020. “Investigating MMM Ponzi scheme on bitcoin.” In *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security,*, 519–530.

Chawla, Nitesh V, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. “SMOTE: synthetic minority over-sampling technique.” *Journal of artificial intelligence research* 16, 321–357.

Chen, Weili, Zibin Zheng, Jiahui Cui, Edith Ngai, Peilin Zheng, and Yuren Zhou. 2018. “Detecting ponzi schemes on ethereum: Towards healthier blockchain technology.” In *Proceedings of the 2018 world wide web conference,*, 1409–1418.

Deason, Stephen, Shivaram Rajgopal, and Gregory B Waymire. 2015. “Who gets swindled in Ponzi schemes?” *Available at SSRN 2586490.*

DeLiema, Marguerite, Yiting Li, and Gary Mottola. 2023. "Correlates of responding to and becoming victimized by fraud: Examining risk factors by scam type." *International Journal of Consumer Studies* 47, 1042–1059.

Deliema, Marguerite, Doug Shadel, and Karla Pak. 2020. "Profiling victims of investment fraud: Mindsets and risky behaviors." *Journal of Consumer Research* 46, 904–914.

Durieux, Thomas, João F Ferreira, Rui Abreu, and Pedro Cruz. 2020. "Empirical review of automated analysis tools on 47,587 ethereum smart contracts." In *Proceedings of the ACM/IEEE 42nd International conference on software engineering*,, 530–541.

Frankel, Tamar. 2012. *The Ponzi scheme puzzle: A history and analysis of con artists and victims.* Oxford University Press.

Gottschalk, Petter. 2017. *Organizational opportunity and deviant behavior: Convenience in white-collar crime.* Edward Elgar Publishing.

Gurun, Umit G, Noah Stoffman, and Scott E Yonker. 2018. "Trust busting: The effect of fraud on investor behavior." *The Review of Financial Studies* 31, 1341–1376.

Guyon, Isabelle, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. 2002. "Gene selection for cancer classification using support vector machines." *Machine learning* 46, 389–422.

Huang, Yuan, Queping Kong, Nan Jia, Xiangping Chen, and Zibin Zheng. 2019. "Recommending differentiated code to support smart contract update." In *Proceedings of the 27th International Conference on Program Comprehension*,, 260–270.

investor.gov. 2022. "Ponzi Scheme." https://www.investor.gov/protect-your-investments/fraud/types-fraud/ponzi-scheme, Last accessed on 2022-06-29.

Krugman, Paul. 2008. "The Madoff Economy." *New York Times* 19.

Lewis, Lionel S. 2013. "The confidence game: of others and of Bernard Madoff." *Society* 50, 283–292.

Lewis, Mervyn K. 2012. “New dogs, old tricks. Why do Ponzi schemes succeed?” In *Accounting Forum*, Volume 36., 294–309, Elsevier.

Lo, T Wing, and Wan Sang Kan. 2023. “How to win trust: The case of P2P financial fraud in China.” *Journal of Criminology* 56, 116–135.

Moore, Tyler, Jie Han, and Richard Clayton. 2012. “The postmodern Ponzi scheme: Empirical analysis of high-yield investment programs.” In *International Conference on financial cryptography and data security*,, 41–56, Springer.

Nguyen, Nhung Thi, An Tuan Nguyen, Ha Thi Nguyet To, and Thanh Ta Hong Le. 2023. “Why are Vietnamese people susceptible to cryptocurrency Ponzi schemes? Findings from using the PLS-SEM approach.” *Journal of Financial Crime.*

Owens, Emily Greene, and Michael Shores. 2010. “Informal networks and white collar crime: Evidence from the Madoff scandal.” *Available at SSRN 1742363.*

Pedregosa, F., G. Varoquaux, A. Gramfort et al. 2011. “Scikit-learn: Machine Learning in Python.” *Journal of Machine Learning Research* 12, 2825–2830.

Rantala, Ville. 2019. “How do investment ideas spread through social interaction? Evidence from a Ponzi scheme.” *The Journal of Finance* 74, 2349–2389.

Raval, Devesh. 2021. “Who is victimized by fraud? Evidence from consumer protection cases.” *Journal of Consumer Policy* 44, 43–72.

Reurink, Arjan. 2018. “Finance Crime.” In *Oxford Research Encyclopedia of Criminology and Criminal Justice.*

Singh, Kamakhya Narain, and Gaurav Misra. 2022. “Victimisation of investors from fraudulent investment schemes and their protection through financial education.” *Journal of Financial Crime.*

Tennant, David. 2011. "Why do people risk exposure to Ponzi schemes? Econometric evidence from Jamaica." *Journal of International Financial Markets, Institutions and Money* 21, 328–346.

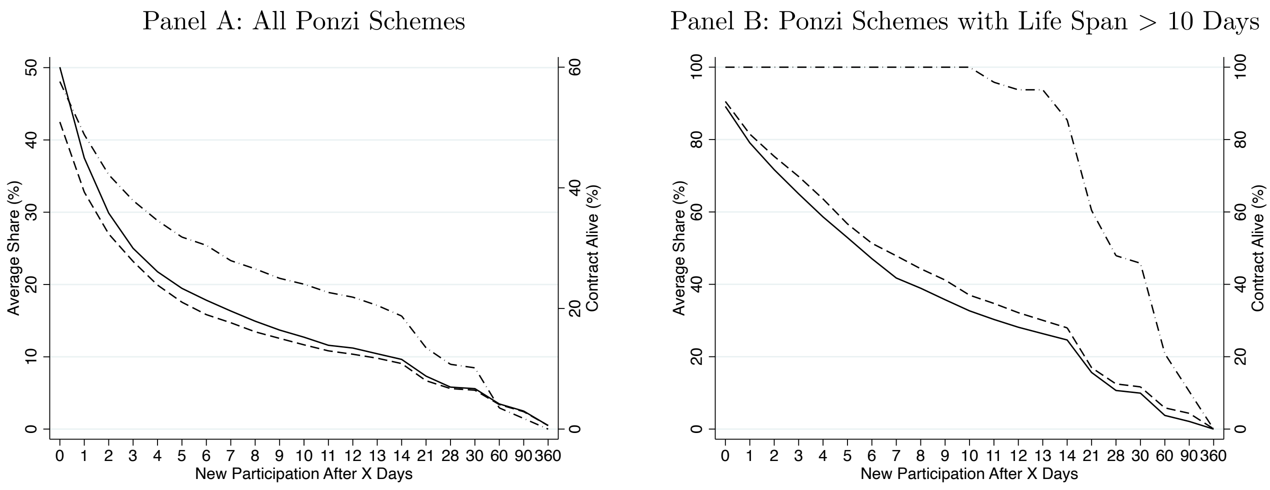


Figure 1: **Ponzi Dynamics over Time: Participants and Investment Shares.** This figure depicts the portion of participants and investment enter the Ponzi game contract after first X calendar days. X takes values of 0, 1 ,2 ,3, 4, 5, 6, 7, 8, 910, 11, 12, 13, 14, 21, 28, 30, 60, 90, 360. Day 0 is the creation day where the Ponzi game smart contract deployed successful on Ethereum. Figure (a) includes all 512 Ponzi schemes, and Figure (b) only includes Ponzi schemes with life span more than 10 days.

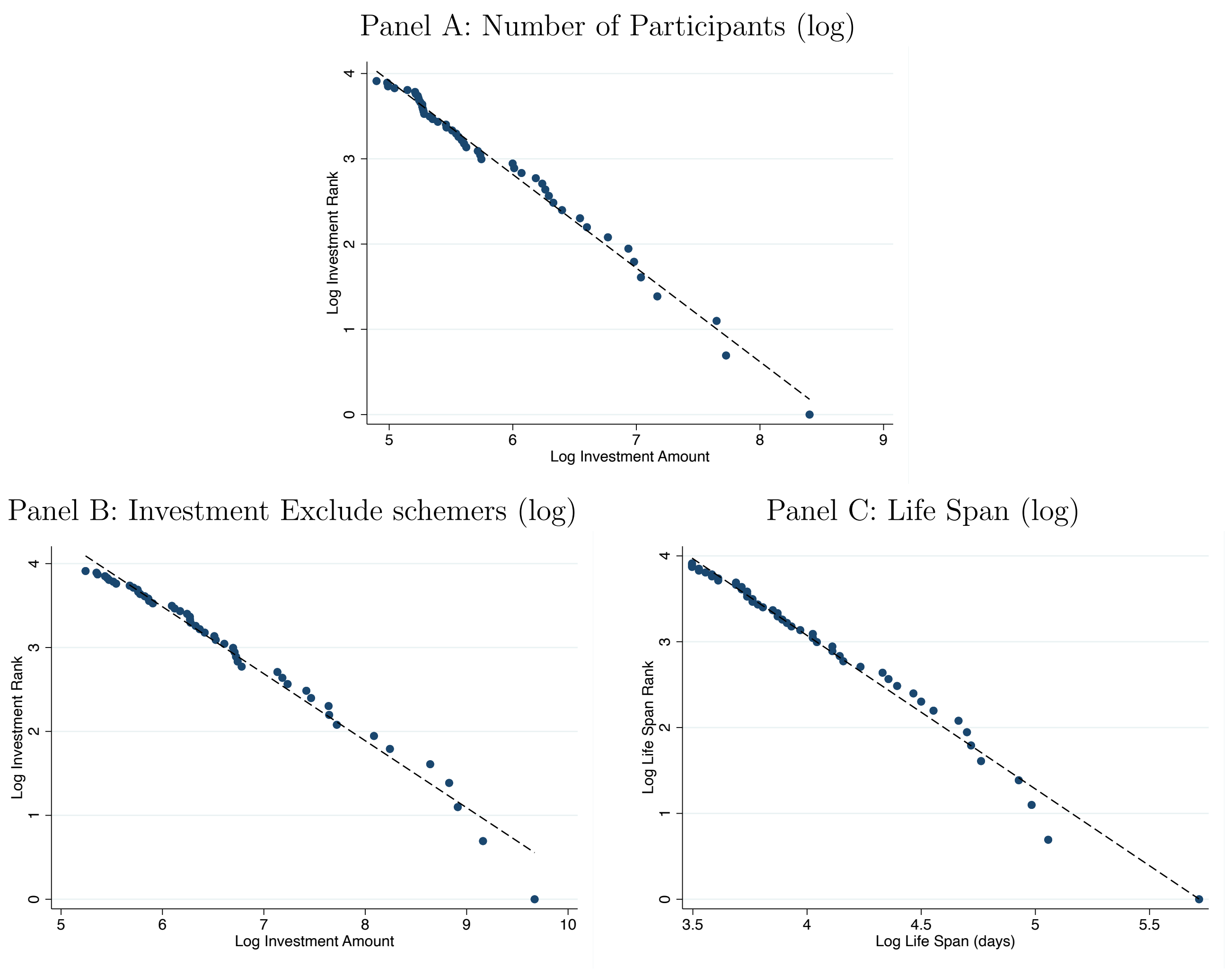


Figure 2: **Zipf's Law of Ponzi Scheme Performance.** This figure plots the Zipf's Law of Ponzi schemes performance measures in the largest 50 contracts (after removing the largest outlier contract: 0x01eacc3ae59ee7fbbc191d63e8e1ccfdac11628c). Each panel plots the log rank and the log number of participants/investment amount/life span. We present the Zipf's law of the log number of addresses in Panel A, the log number of ETH investment amount in Panel B, and the log life span in Panel C. We estimate the following regressions:

$$log(Performance\_Rank_i) = log(Performance_i) + \gamma + \epsilon_i$$

The slope in Panel A is -0.799 (s.e.=0.016) and R-squared is 98.09%, the slope in Panel B is -1.098 (s.e.=0.016) and R-squared is 99.02%, and the slope in Panel C is -1.788 (s.e.=0.031) and R-squared is 98.62%.

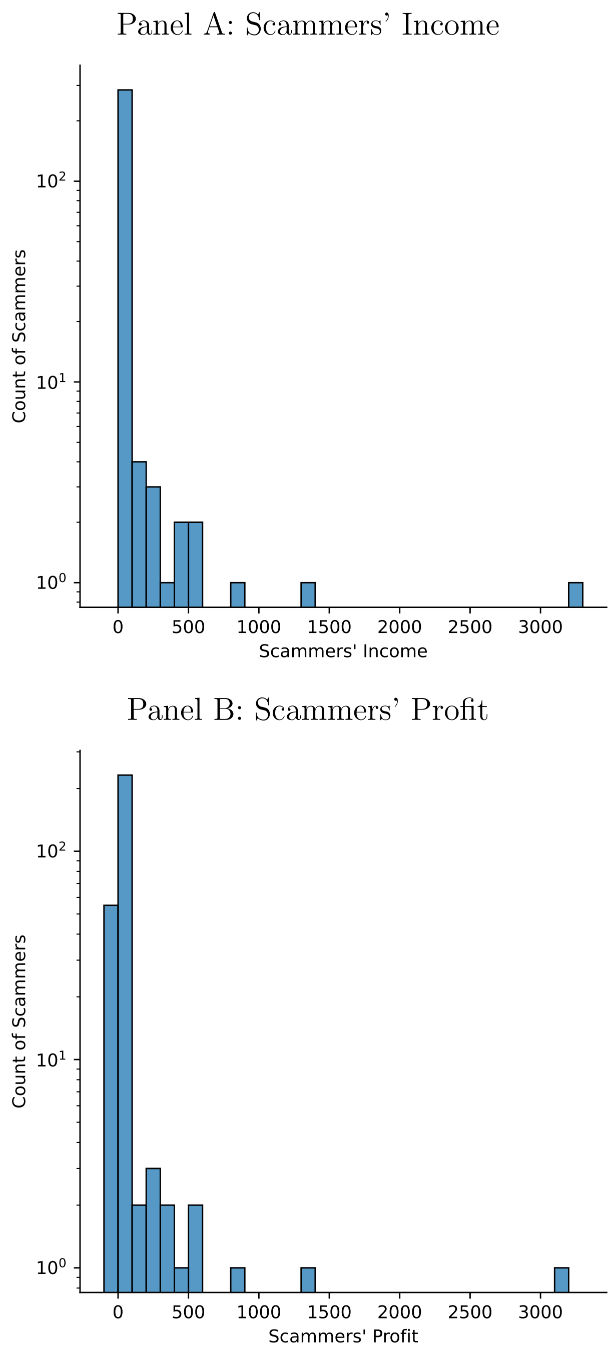


Figure 3: **Histogram of Income and Profit of Scammers** This figure depicts the distribution of income and profit of scammers of Ponzi scheme. Scammers include the creators of Ponzi contract and the fee address registered in the contract. We calculate the profit by following regressions:

$$Profit_i = Income_i - Investment_i$$

To show the Histogram clearly, the observation which have 27617.08 ETH income and 24934.17 ETH Profit is not plotted in the figure

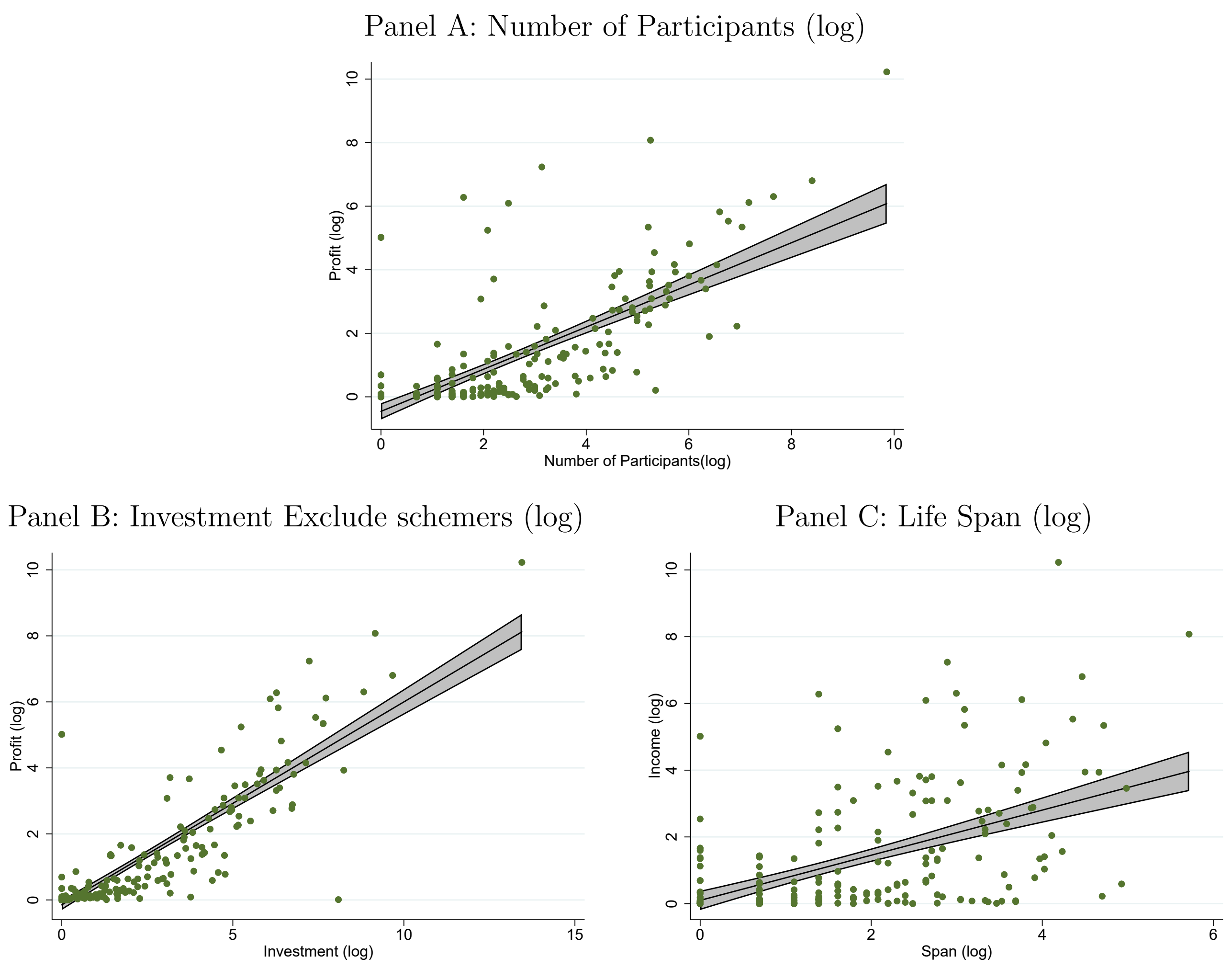


Figure 4: **Profit and Expenditure of Scammer** This figure describes the profit of scammers of Ponzi scheme and Ponzi size. There are 276 schemes which scammer account have transaction with the Ponzi contract among all 512 Ponzi contract. Among the 276 schemes, only 187 schemes' scammer make profit while other 89 contract scammers lost investment in thier own schemes. In this figure, we plot the 187 ponzi where scammer's make profit.

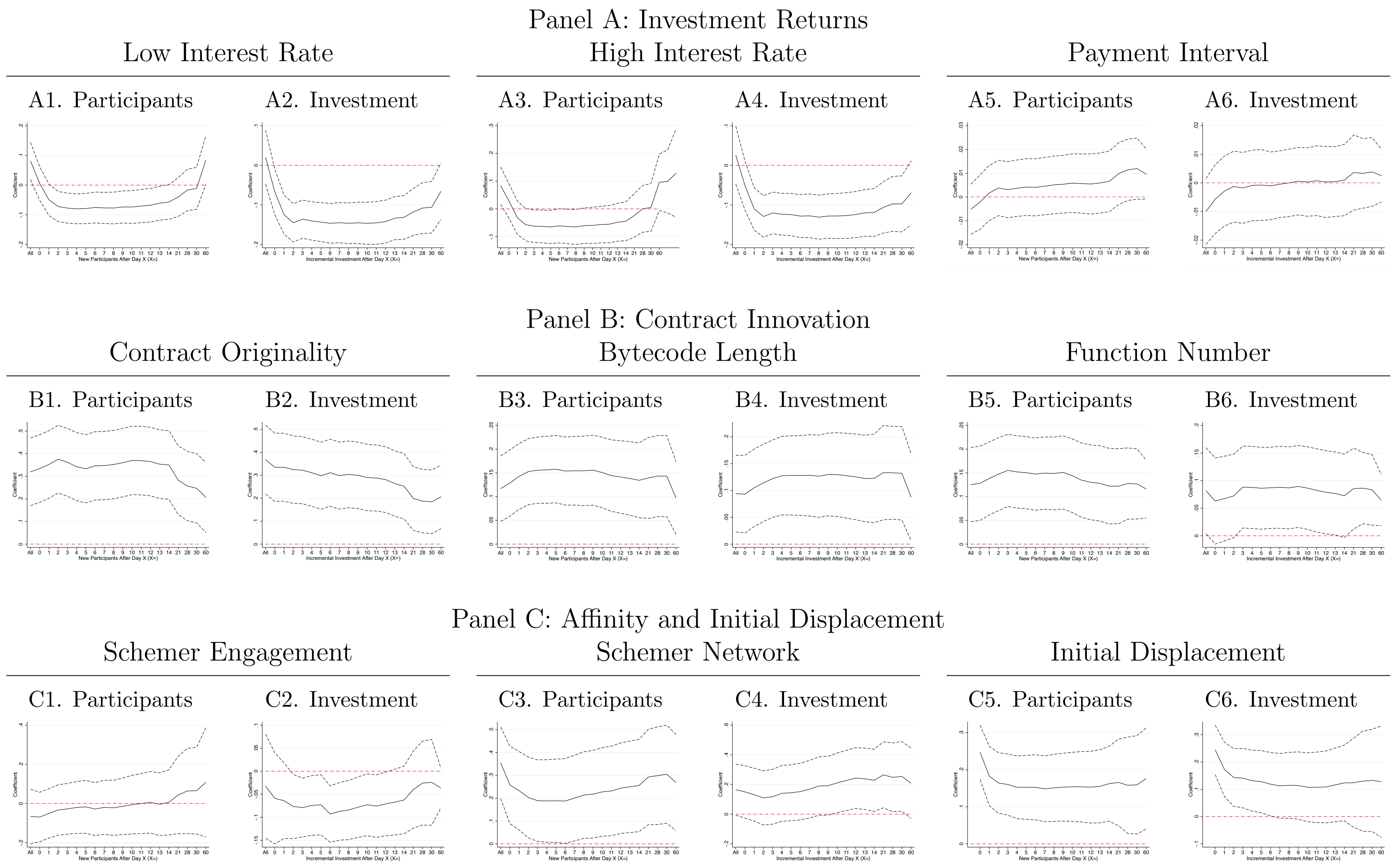


Figure 5: **Dynamic Effect of Determinants.** This figure plots the dynamic effects of the determinants. $Participant^{norm}_{i,t}$, the normalized (mean zero, s.d. one) log new participants (wallet addresses) after Day t since the smart contract being deployed. In Panels A1, A3, B1, B3, B5, C1, C3, and C5, we regress $Participant^{norm}_{i,t}$ on Ponzi game feature i ($Feature_i$) and plot coefficients β_t as a function of t:

$$log(Participant^{norm}_{i,t}) = \beta_t Feature_i + \gamma + \epsilon_{i,t}$$

$Invest^{norm}_{i,t}$ refers to the normalized (mean zero, s.d. one) log new investment amount (in ETH) after Day t since the smart contract being deployed. In Panels A2, A4, B2, B4, B6, C2, C4, and C6, we regress $Invest^{norm}_{i,t}$ on Ponzi game feature i ($Feature_i$) and plot coefficients β_t as a function of t:

$$log(Invest^{norm}_{i,t}) = \beta Feature_i + \gamma + \epsilon_{i,t}$$

Dash lines are 90% confidential intervals based on robust standard errors, and the red vertical dash dotted lines benchmark $\beta = 0$.

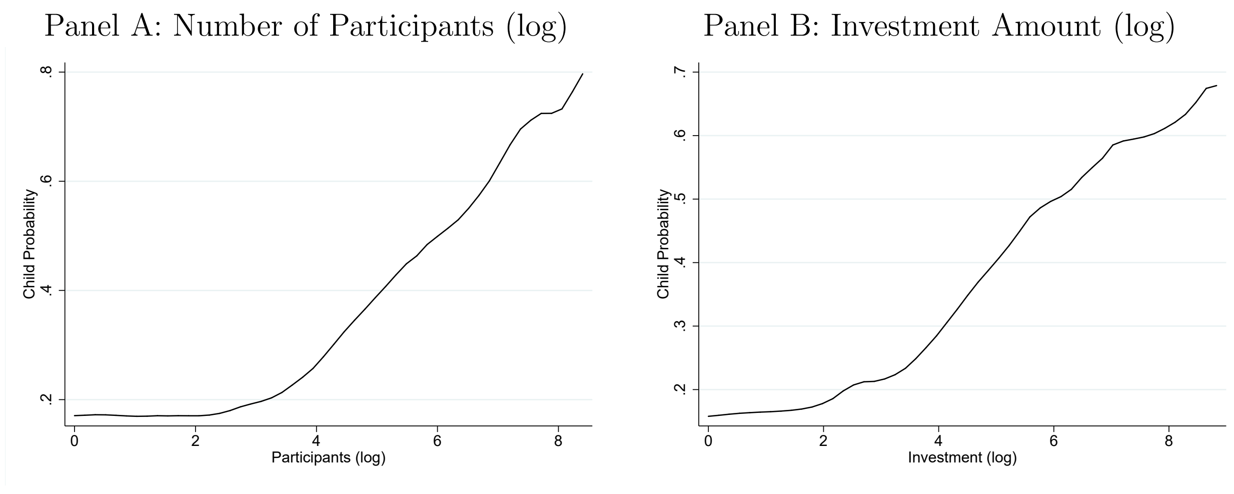


Figure 6: **Child Probability (clustered by functions)** This figure plots the relation between the performance of Smart Ponzi and probality of original smart contract have copycats (child). The X-ray represent the performance of Smart Ponzi , while the Y-ray represents the probability of having child. The probability is defined as the percentage of Smart Ponzi with child in the group of contracts have same performance. Panel A use number of participants the as proxy of Smart Ponzi performance. Panel B use investment amount the as proxy of Smart Ponzi performance.

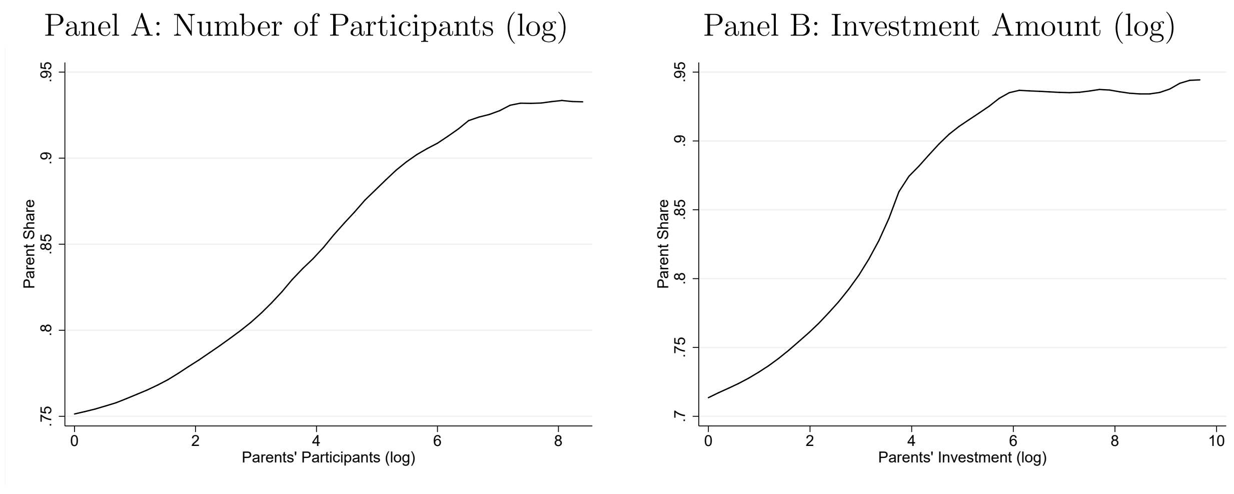


Figure 7: **Parent Share (clustered by functions)** This figure plots the relation between the share of original smart in the group include original smart and their child. The X-ray represent the performance of parents of Smart Ponzi , while the share of parents. Panel A use number of participants the as proxy of Smart Ponzi performance. Panel B use investment amount the as proxy of Smart Ponzi performance.

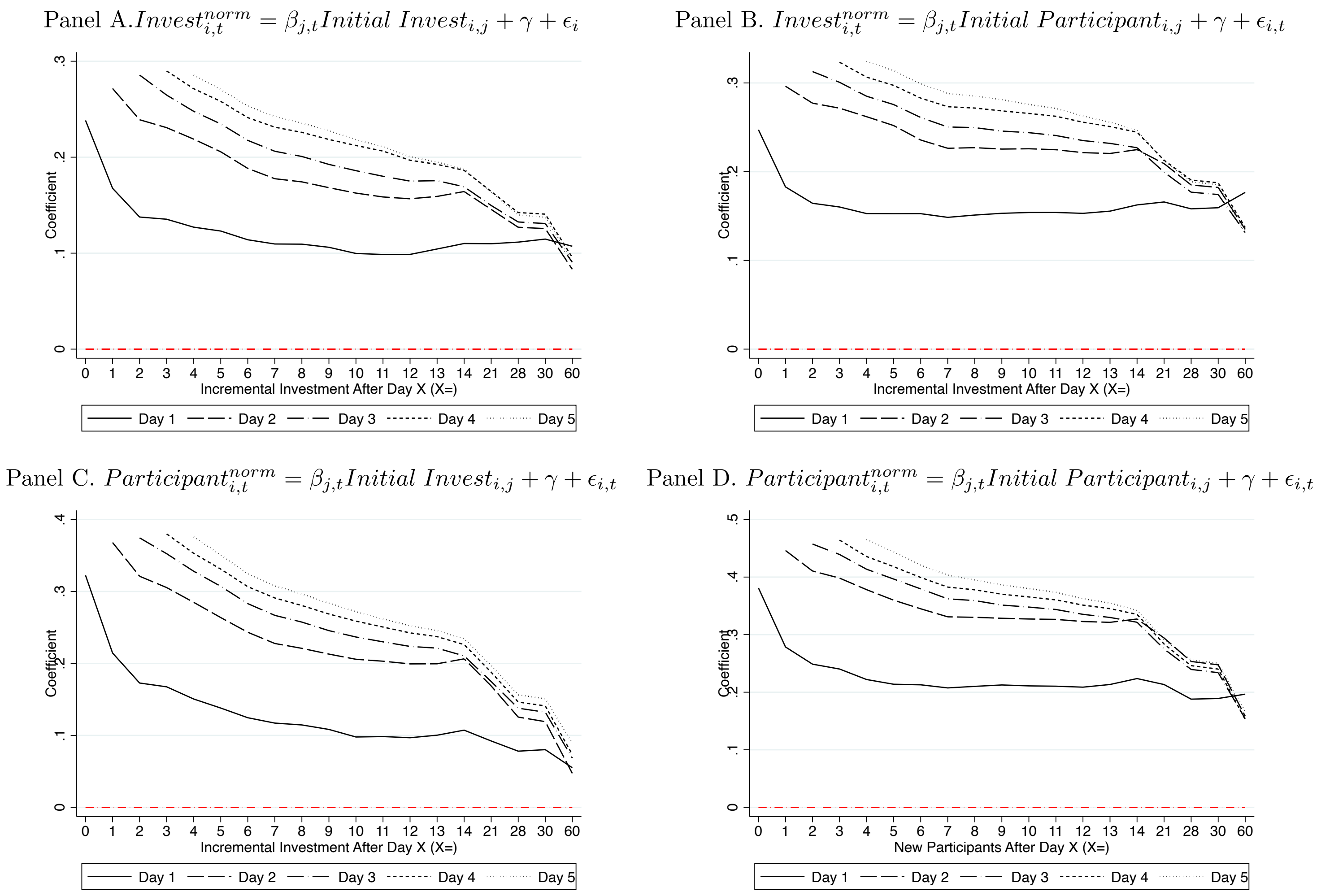


Figure 8: **Predict Future Engagement with Initial Displacement.** This figure plots the predictability of further engagement after Day j(new participants and incremental investment) of the initial participation in the first j days. $Invest^{norm}_{i,t}$ refers to the normalized (mean zero, s.d. one) log new investment amount (in ETH) after Day t since the smart contract being deployed , and $Participant^{norm}_{i,t}$ refers to the normalized (mean zero, s.d. one) log new participants (wallet addresses) after Day t since the smart contract being deployed. $Initial\ Invest_{i,j}$ refers to the log total investment amount (in ETH) in the first j days, $Initial\ Participant_{i,j}$ refers to the log total number of participants (wallet address) in the first j days. In each figure, we plot $\beta_{j,t}$ $(j = 1, 2, 3, 4, 5)$ as a function of t where $t \geq j$ to avoid overlapping in time. The red vertical dash dotted lines benchmark $\beta = 0$.

Panel A: Number of Participants (log) Panel B: Investment Exclude Schemers (log)

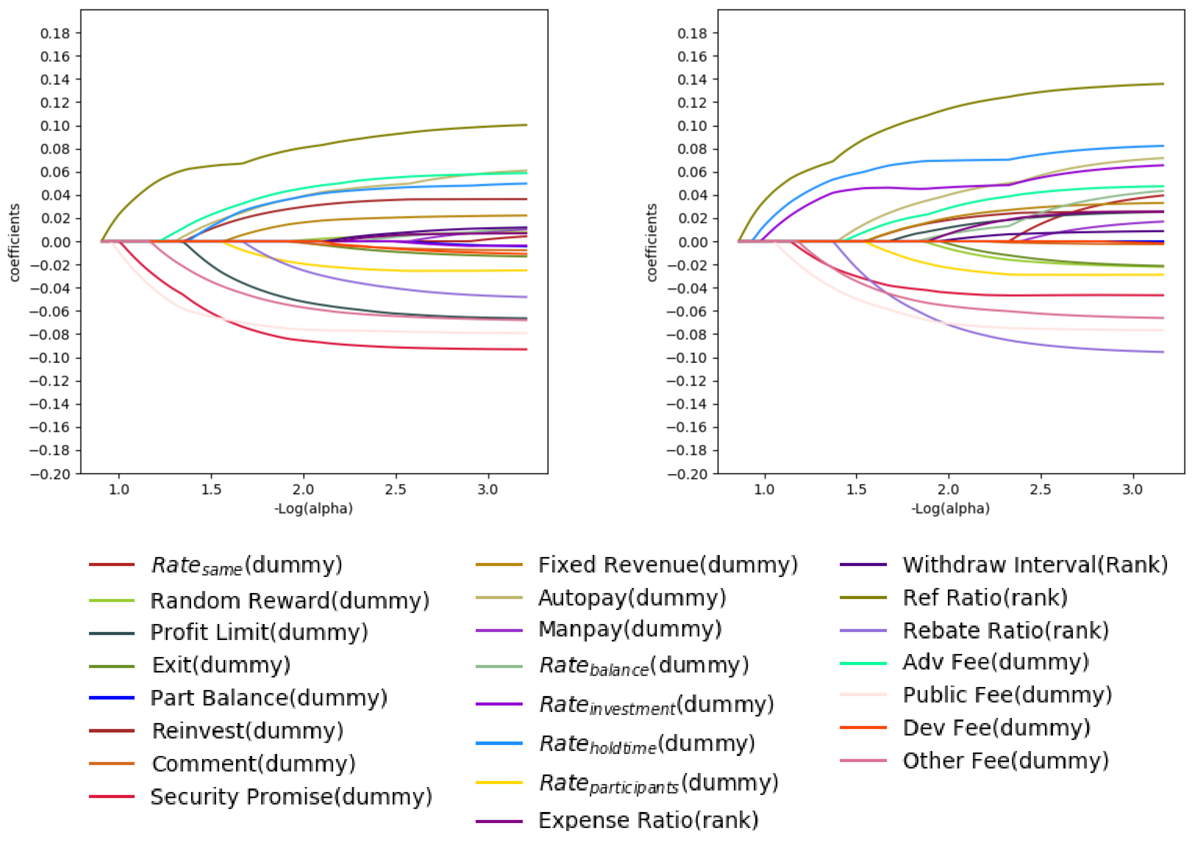
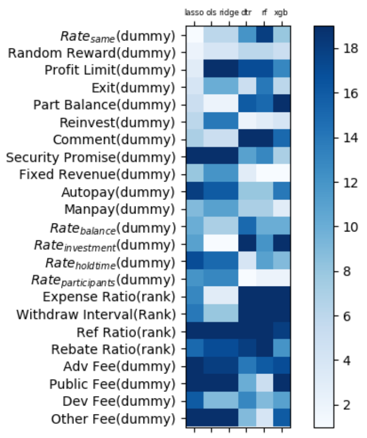


Figure 9: **Feature Selection by Lasso** This figure plots the Lasso Path of different variables in the predicting the size of Smart Ponzis. The LASSO regression minimizes the following objective function:

$$\min_{w} \frac{1}{2n_{\text{samples}}} \|Xw - y\|_2^2 + \alpha \|w\|_1 \tag{3}$$

where the α is a constant and $\|w\|_1$ is the l1-norm of the coefficient vector. The X-ray represent the different values of parameter α , while the Y-ray represents the coefficients β estimated by the α. The line is the regularization path by using different regularization parameter. Panel A use number of participants the as proxy of Smart Ponzi size. Panel B use investment amount exclude schemers the as proxy of Smart Ponzi size.

Panel A: Number of Participants (log)



Panel B: Investment Exclude Schemers (log)

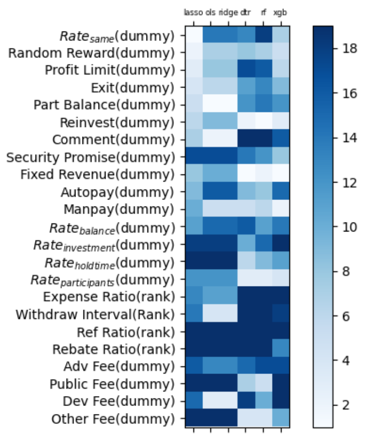


Figure 10: **Distribution of Ponzi Scheme Performance.** This figure plots the ranking of different variables in the predicting the size of Smart Ponzis. The column represent the features' rank of importance in different models. To measure the importance in different models, we apply the RFE method (Recursive feature elimination). Panel A use number of participants as proxy of Smart Ponzi size. Panel B use investment amount exclude schemers as proxy of Smart Ponzi size. Panel D uses life span the as proxy of Smart Ponzi size.

Table 1: **Criteria for Different Categories**

	C1	C2	C3	C4	C5
Ponzi	✓	✓	✓	✓	✓
Gambling	✓	✓		✓	✓
Fomo3D	✓	✓		✓	
ICO	✓	✓			✓
Token		✓			
Tokensale		✓			

Notes: This table reports how many criteria are fulfilled in different type of Smart Contracts. C1 to C5 refer to different criteria. We select 6 different type of Smart Contracts. A check mark shows that the criterion is fulfilled.

Table 2: **Schemer's Affinity and Profit**

Panel A: User's Profit (Dummy)						
	(1)	(2)	(3)	(4)	(5)	(6)
Isfriend_c_neat	-0.058	-0.071			-0.053	0.061
	(0.045)	(0.057)			(0.045)	(0.055)
Isfeecreator			0.442***	0.611***	0.442***	0.613***
			(0.018)	(0.023)	(0.018)	(0.023)
Constant	0.331***	0.331***	0.326***	0.324***	0.326***	0.324***
	(0.002)	(0.000)	(0.002)	(0.000)	(0.002)	(0.000)
Contract Cluster	N	Y	N	Y	N	Y
R-squared	0.000	0.193	0.010	0.204	0.010	0.204
Panel B: User's Profit						
	(1)	(2)	(3)	(4)	(5)	(6)
Isfriend_c_neat	0.065	-0.114			0.763***	22.523
	(0.503)	(0.098)			(0.116)	(16.197)
Isfeecreator			63.680	104.328	63.682	105.346
			(44.556)	(73.220)	(44.557)	(73.933)
Constant	-0.198	-0.198***	-0.895***	-1.340*	-0.896***	-1.394*
	(0.496)	(0.000)	(0.080)	(0.801)	(0.080)	(0.840)
Contract Cluster	N	Y	N	Y	N	Y
R-squared	0.000	0.002	0.003	0.008	0.003	0.008
N	51630	51630	51630	51630	51630	51630

Notes: This table evaluates schemer and their affinity network and their profit from the Ponzi Scheme. In panel A, we use a dummy variable whether user make profit from the scheme as the dependent variable. In panel B, we use the user's Profit the dependent variable. Robust Standard errors are reported in parentheses,* $p < 0.1$, ** $p < 0.05$, *** $p < 0.01$.

Table 3: **Summary Statistics**

	Count	Mean	sd	Min	Median	Max
Panel A: Size of Ponzi Scheme						
Participants	512	100.84	883.44	1.00	4.00	19069.00
Investment amount	512	1527.32	30641.95	0.00	0.58	693146.56
Life Span	512	37.86	103.70	0.00	2.00	723.00
Panel B: Contract Complexity						
Function Numbers	512	14.39	11.01	1.00	13.00	45.00
Bytecode Length	512	7461.48	8215.63	436.00	4563.00	55740.00
Panel C: Contract Originality						
$Original_{name}(dummy)$	512	0.49	0.50	0.00	0.00	1.00
$Parentnumber_{name}(dummy)$	263	1.17	0.54	1.00	1.00	4.00
$ChildAbility_{name}(dummy)$	512	0.19	0.39	0.00	0.00	1.00
$Original_{fun}(dummy)$	512	0.45	0.50	0.00	0.00	1.00
$Parentnumber_{fun}(dummy)$	284	1.27	0.97	1.00	1.00	6.00
$ChildAbility_{fun}(dummy)$	512	0.17	0.38	0.00	0.00	1.00
Panel D: Interest Settings						
Lowest Interest	512	23.95	164.80	0.00	4.00	2400.00
Highest interest	486	26.22	168.67	0.00	5.00	2400.00
Panel E: Schemers's Network, Initial Displacement Investment and Creator's Engagement						
Scammer Investment	512	1.50	12.54	0.00	0.00	150.00
Initial Investment	512	1511.37	30513.81	0.00	0.11	690236.56
Acc TX	512	47.39	231.55	0.00	1.00	2272.00
Acc Life	512	32.99	81.19	0.00	1.00	469.00
Connected Acc	512	8.40	33.60	0.00	1.00	529.00
Invest Acc	512	0.25	0.71	0.00	0.00	8.00
Share	276	0.08	0.18	0.00	0.00	1.00

Notes: This table reports the summary statistics of variables used for analysis. Panel A tabulates the three metrics of the Ponzi game size. Panel B summarizes the two variables of contract complexity, Panel C summarizes contract originality identified by function cluster and contract names, Panel D summarizes the range of daily interest rate of Ponzi games, Panel E summarizes the variables related to the schemers's affinity and early investment in Ponzi.

Table 4: **Function Originality**

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)
	Success		Log(Participants)		Log(Investment)		log(Life)		Scammers' Profit	
Original_Function	0.108***	0.227***	0.603***	1.205***	0.784***	1.570***	0.376***	0.537*	1.044***	1.431**
	(0.027)	(0.068)	(0.172)	(0.408)	(0.207)	(0.521)	(0.126)	(0.290)	(0.273)	(0.644)
Function Cluster	N	Y	N	Y	N	Y	N	Y	N	Y
N	512	512	512	512	512	512	512	512	187	187
R^2	0.034	0.449	0.025	0.449	0.030	0.441	0.018	0.430	0.006	0.996

Notes: For each contract i in function cluster j, we run the following regressions with and without function cluster dummies:

$$Performance_i = \beta Origin\_Function_{i,j} + \gamma_j + \epsilon_i$$

$Performance_i$ takes the success dummy (survive more than 10 days, and more than 100 wallet addresses engaged) in Columns (1) and (2), log number of participants (wallet addresses) in Columns (3) and (4), log total amount of investment in Columns (5) and (6), log life span in Columns (7) and (8) and scammers' profit from Ponzi in Columns (9) and (10). $Origin\_Function_{i,j} = 1$ if the contract i is the first-deployed one in a function cluster. Otherwise, $Origin\_Function_{i,j} = 0$. Function cluster dummies are excluded in Columns (1), (3), (5), and (7), and are included in Columns (2), (4), (6), and (8). The robust standard errors are clustered at the function group and reported in parentheses. * $p < 0.1$, ** $p < 0.05$, *** $p < 0.01$.

Table 5: **Innovation: New Contract Name or Function Originality**

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
	Success		Log(Participants)		Log(Investment)		Log(Life)	
Origin_Name	0.021	-0.012	0.166	-0.015	0.047	-0.205	0.082	-0.033
	(0.026)	(0.025)	(0.167)	(0.169)	(0.202)	(0.200)	(0.125)	(0.132)
Origin_Function		0.111***		0.607***		0.846***		0.386***
		(0.027)		(0.176)		(0.209)		(0.134)
N	512	512	512	512	512	512	512	512
R^2	0.001	0.034	0.002	0.025	0.000	0.031	0.001	0.018

Notes: This table compares the seed contract identified by contract names with the seed contract identified by function clusters in predicting the performance of Ponzi games.

$$Performance_i = \beta_1 Origin\_Name_{i,j'} + \beta_2 Origin\_Function_{i,j} + \gamma + \epsilon_i$$

$Performance_i$ takes the success dummy (survive more than 10 days, and more than 100 wallet addresses engaged) in Columns (1) and (2), log number of participants (wallet addresses) in Columns (3) and (4), log total amount of investment in Columns (5) and (6), and log life span (the number of days between the first investment and last investment day) in Columns (7) and (8). $Origin\_Function_{i,j} = 1$ if the contract i is the first-deployed one in a function cluster. Otherwise, $Origin\_Function_{i,j} = 0$. $Origin\_Name_{i,j} = 1$ if the contract i is the first contract in the group sharing similar names. Otherwise, $Origin\_Name_{i,j} = 0$. Robust Standard errors are reported in parentheses,* $p < 0.1$, ** $p < 0.05$, *** $p < 0.01$.

Table 6: **Correlation Between Child's Size and Parent Size**

Panel A: Mean Size of Different Clusters			
	$Participants_{parent}$ (log)	$Invest_{parent}$ (log)	$Span_{parent}$
$Participants_{child}$ (log)	**0.172**	**0.162**	0.056
$Invest_{child}$ (log)	0.099	0.124	0.021
$Span_{child}$	0.088	0.075	**0.186***
Panel B: Max Size in Each Cluster			
	$Participants_{parent}$ (log)	$Invest_{parent}$ (log)	$Span_{parent}$
$Participants_{child}$ (log)	**0.223***	**0.211***	0.110
$Invest_{child}$ (log)	**0.165**	**0.182**	0.078
$Span_{child}$	**0.230***	**0.181**	**0.273***
Panel C: Sum Size of Different Clusters			
	$Participants_{parent}$ (log)	$Invest_{parent}$ (log)	$Span_{parent}$
$Participants_{child}$ (log)	**0.247***	**0.236***	**0.143**
$Invest_{child}$ (log)	**0.181**	**0.197***	0.108
$Span_{child}$	**0.267***	**0.217***	**0.373***

Notes: This table reports the correlation between smart Ponzi contracts' size and their childs' size. We use the BOW(bag of words) model to encode the ABI of smart contract into the sparse vector. Then, we use Birch algorithm to calculate the similarity of different Smart Ponzi Contracts and cluster them. Finally, we got 277 clusters of smart Ponzis. In each cluster, we define the Smart Ponzi created at the earliest date as parent, else are labeled as child. We labeled 228 contracts as parent while 284 contracts are labeled as child. Only 89 clusters have childs, so we use these 89 clusters to analyse correlation. In Panel A, we calculated the average size of parents and average size of childs in each clusters to measure the size of parents and childs. In Panel B, we use the most successful contract to measure the size of parents and childs by selecting the maximum size of parents and childs in each cluster. In Panel C, we use the cumulative size to measure the size of parents and childs by calculating the sum of size in each clusters. Correlations with 5% significance are in bold. Correlations with 1% significance are *.

Table 7: **Contract Complexity**

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
	Success		Log(Participants)		Log(Investment)		Life	
Log(Function Number)	0.028**		0.236***		0.186*		0.151**	
	(0.014)		(0.089)		(0.108)		(0.068)	
Log(Bytecode Length)		0.025**		0.220***		0.214**		0.186***
		(0.011)		(0.079)		(0.099)		(0.058)
N	512	512	512	512	512	512	512	512
R^2	0.007	0.009	0.012	0.016	0.007	0.011	0.009	0.021

Notes: This table evaluates the importance of contract complexity.

$$Performance_i = \beta Log(Function\ Number_i)/Log(Bytecode\ Length_i) + \gamma + \epsilon_i$$

The contract complexity is measured by log number of function numbers ($Log(Function\ Number_i)$) in Columns (1), (3), (5), and (7); measured by log bytecode length of the smart contract in Columns (2), (4), (6), and (8). $Performance_i$ takes the success dummy (survive more than 10 days, and more than 100 wallet addresses engaged) in Columns (1) and (2), log number of participants (wallet addresses) in Columns (3) and (4), log total amount of investment in Columns (5) and (6), and log life span (the number of days between the first investment and last investment day) in Columns (7) and (8). Robust Standard errors are reported in parentheses,* $p < 0.1$, ** $p < 0.05$, *** $p < 0.01$.

Table 8: **Schemer's Experience and Affinity Fraud**

	Creator Account's Experience			Affinity		
	(1)	(2)	(3)	(4)	(5)	(6)
Panel A: Log (Number of Participants)						
Indep Var	Log(TX)	Log(Acc Life)	Log(Connected Acc)	Log(Invest Acc)	Share	Share
Coef.	-0.038	-0.038	-0.061	0.209**	6.721**	14.683***
s.e.	(0.051)	(0.043)	(0.059)	(0.098)	(2.994)	(4.695)
R-squared	0.001	0.001	0.002	0.006	0.073	0.145
N	512	512	512	512	118	71
Panel B: Log (Total Investment)						
Indep Var	Log(TX)	Log(Acc Life)	Log(Connected Acc)	Log(Invest Acc)	Share	Share
Coef.	-0.040	-0.010	-0.085	0.093	7.239	13.702***
s.e.	(0.066)	(0.052)	(0.067)	(0.128)	(4.504)	(5.010)
R-squared	0.001	0.000	0.002	0.001	0.053	0.106
N	512	512	512	512	118	71
Panel C: Log (Life Span)						
Indep Var	Log(TX)	Log(Acc Life)	Log(Connected Acc)	Log(Invest Acc)	Share	Share
Coef.	0.001	0.001	-0.003	0.204**	3.639**	12.491***
s.e.	(0.037)	(0.034)	(0.049)	(0.090)	(1.549)	(4.066)
R-squared	0.000	0.000	0.000	0.011	0.038	0.163
N	512	512	512	512	118	71
Panel D: Log(Scammers' Income)						
Indep Var	Log(TX)	Log(Acc Life)	Log(Connected Acc)	Log(Invest Acc)	Share	Share
Coef.	0.028	0.015	0.007	0.033	4.439	5.259
s.e.	(0.044)	(0.031)	(0.036)	(0.087)	(3.690)	(3.785)
R-squared	0.001	0.000	0.000	0.000	0.052	0.046
N	512	512	512	512	118	71
Panel E: Log(Scammers' Profit)						
Indep Var	Log(TX)	Log(Acc Life)	Log(Connected Acc)	Log(Invest Acc)	Share	Share
Coef.	0.066	0.079	-0.005	-0.064	5.672	7.678
s.e.	(0.110)	(0.085)	(0.091)	(0.206)	(5.913)	(8.172)
R-squared	0.004	0.005	0.000	0.001	0.060	0.047
N	187	187	187	187	49	28

Notes: This table evaluates schemer's account experience and affinity network.

$$Performance_i = \beta Feature_i + \gamma + \epsilon_i$$

$Performance_i$ is number of participants of contract i in Panel A, number of total investment in ETH in Panel B, life span in Panel C, scammers' income in Panel D and scammers' profit in Panel E. Columns (4)-(6) measure the experience of creators while columns (4)-(6) measure the affinity. $Feature_i$ takes the log number of transactions made by Ponzi schemer's address in Column (1), log account life (number of days between the address creation date and the Ponzi game creation date) in Column (2), and log number of connected account (number of addresses execute transactions with the schemer's experience) in Column (3). Columns (1)-(3) measure the schemer account's experience. $Feature_i$ takes log number of connected accounts which invest in the Ponzi game in Column (4), shares of connected accounts which invest in the Ponzi out of total connected accounts in Panel A-C Column (5) conditional 118 contracts with at least 5 connected accounts, shares of connected accounts which invest in the Ponzi out of total connected accounts in Column (6) conditional 71 contracts with at least 10 connected accounts.In panel E, we conduct a regression analysis on the subset of observations where scammers' profits are greater than zero. Robust Standard errors are reported in parentheses,* $p < 0.1$, ** $p < 0.05$, *** $p < 0.01$.

Table 9: **Investors Experience and Ponzi Expansion**

	(1)	(2)	(3)	(4)	(5)	(6)
	Panel A: Log (Number of Participants After N Days)					
Dep Var	All ($N = 0$)		$N = 3$ Days		$N = 5$ Days	
Indep Var	Life Day 1	TX Day 1	Life Day 3	TX Day 3	Life Day 5	TX Day 5
Coef.	0.197***	0.224***	0.274***	0.226***	0.260***	0.251***
s.e.	(0.040)	(0.042)	(0.033)	(0.045)	(0.030)	(0.041)
R-squared	0.047	0.051	0.079	0.059	0.078	0.090
N	512	512	512	512	512	512
	Panel B: Log (Investment Amount (in ETH))					
Dep Var	All ($N = 0$)		$N = 3$ Days		$N = 5$ Days	
Indep Var	Life Day 1	TX Day 1	Life Day 3	TX Day 3	Life Day 5	TX Day 5
Coef.	0.186***	0.195***	0.218***	0.165***	0.206***	0.196***
s.e.	(0.047)	(0.051)	(0.038)	(0.053)	(0.035)	(0.049)
R-squared	0.029	0.027	0.039	0.024	0.038	0.042
N	512	512	512	512	512	512
	Panel C: Log (Life Span)					
Dep Var	All					
Indep Var	Life Day 1	TX Day 1	Life Day 3	TX Day 3	Life Day 5	TX Day 5
Coef.	0.050*	0.039	0.246***	0.165***	0.277***	0.211***
s.e.	(0.030)	(0.033)	(0.029)	(0.031)	(0.027)	(0.028)
R-squared	0.005	0.003	0.106	0.051	0.131	0.093
N	512	512	512	512	512	512
	Panel D: Log(Scammers' Profit (in ETH))					
Dep Var	All ($N = 0$)		$N = 3$ Days		$N = 5$ Days	
Indep Var	Life Day 1	TX Day 1	Life Day 3	TX Day 3	Life Day 5	TX Day 5
Coef.	0.088	0.102*	0.170**	0.198**	0.167**	0.191**
s.e.	(0.062)	(0.057)	(0.072)	(0.078)	(0.071)	(0.075)
R-squared	0.009	0.012	0.020	0.044	0.019	0.047
N	187	187	187	187	187	187

Notes: This table evaluates schemer's account experience and affinity network.

$$Performance_i = \beta Feature_i + \gamma + \epsilon_i$$

$Performance_i$ is log number of participants of contract i in Panel A, log number of total investment in ETH in Panel B, , log life span (number of days from the first investment to the last investment) in Panel C and log Scammers' Profit in ETH in Panel D. In Panels A, B and D, dependent variables use the all investment amount and total participants in Columns (1) and (2); incremental investment amount and new participants after the first 3 days since contract deployment in Columns (3) and (4); incremental investment amount and new participants after the first 5 days since contract deployment in Columns (5) and (6). $Feature_i$ takes log average account life of all participants in the first one/three/five days in Columns (1), (3), and (5) respectively; log average number of transactions in the first one/three/five days in Columns (2), (4), and (6) respectively. In panel D, only the observations that scammers' profit larger than 0 are in analysis. Robust Standard errors are reported in parentheses,* $p < 0.1$, ** $p < 0.05$, *** $p < 0.01$.

Table 10: **Horse Racing of Size**

	(1) Participants (log)	(2) Exclude (log)	(3) Life (log)	(4) Income (log)	(5) Participants (log)	(6) Exclude (log)	(7) Life (log)	(8) Income (log)
$Rate_{same}$(d)							-0.028	
$Rate_{investment}$(d)				0.128*		0.057		0.133
				(0.076)				
$Rate_{holdtime}$(d)						0.071		
$Rate_{participants}$(d)								
Ref Ratio(r)		0.139**		0.154**	0.095	0.131	0.040	0.124
		(0.066)		(0.074)				
Rebate Ratio(r)		-0.096*		-0.099*		-0.079		
		(0.057)		(0.059)				
Random Reward(d)							0.036	
Security Promise(d)	-0.094*				-0.084	-0.055		-0.076
	(0.052)							
Public Fee(d)	-0.079***	-0.077***	-0.035***	-0.047**	-0.101	-0.080		
	(0.024)	(0.022)	(0.012)	(0.021)				
Adv Fee(d)							0.055	
Dev Fee(d)								0.042
Other Fee(d)	-0.063*	-0.063***	-0.035**		-0.067	-0.069		
	(0.033)	(0.024)	(0.017)					
Reinvest(d)	0.036***	0.026**		0.028**				
	(0.013)	(0.012)		(0.014)				
Fixed Revenue(d)		0.038*						
		(0.020)						
Autopay(d)			0.069*				0.039	
			(0.041)					
Withdraw Interval(r)								0.075
N	512	512	512	512	512	512	512	512

Notes: This table presents the OLS and LASSO regression results for different variables. The dependent variables represent different measures of performance of Smart Ponzi and scammers' income. Participants refers to the number of addresses that invest in the Smart Ponzi. Exclude refers to the total amount raised by the Smart Ponzi excluding the schemer address' investment. Life span refers to the time period between contract creation and the last transaction. Income refers to the scammers' income from Smart Ponzi. All dependent variables are in logarithmic form. All variables are normalized. Columns (1)-(4) show the OLS regression results, where variables with p-value greater than 0.1 are omitted. Columns (5)-(8) show the LASSO regression results, where variables with coefficients equal to zero are omitted. Robust standard errors are in parentheses, * $p < 0.1$, ** $p < 0.05$, *** $p < 0.01$.

For Online Publication

June 18, 2023

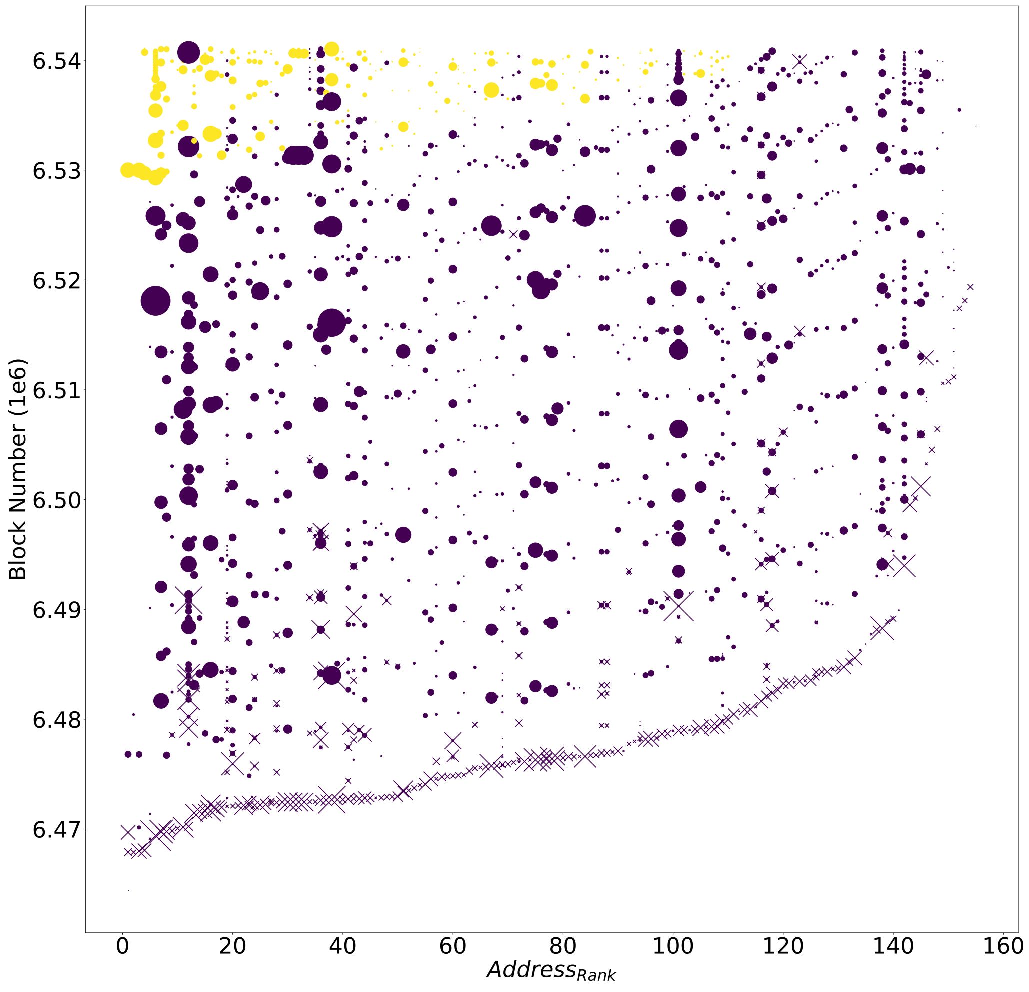


Figure IA1: **A Typical Ponzi Scheme.** This figure depicts a typical smart Ponzi scheme. Each point represents a transaction. One column of the point represents all transactions made by an investor with Smart Ponzi. On the horizontal axis, participants are arranged in order of participation time. The vertical axis represents the block number where the transaction took place. Different shapes refer to different actions. A cross refers to a transaction the investor deposits in Smart Ponzi, while the circle refers to payment from the smart contract to users who interacted with the contract. If the investor is break even, the color of the action is yellow while the color is purple. The size of the scatter represents the value of the transaction. At the end of the Smart Ponzi, many existing investors cashed out.

```
/**
 *Submitted for verification at Etherscan.io on 2018-10-07
*/
pragma solidity ^0.4.24;
contract EasyInvest10 {
    // records amounts invested
    mapping (address => uint256) public invested;
    // records blocks at which investments were made
    mapping (address => uint256) public atBlock;
    // this function called every time anyone sends a transaction to this contract
    function () external payable {
        // if sender (aka YOU) is invested more than 0 ether
        if (invested[msg.sender] != 0) {
            // calculate profit amount as such:
            // amount = (amount invested) * 10% * (blocks since last transaction) / 5900
            // 5900 is an average block count per day produced by Ethereum blockchain
            uint256 amount = invested[msg.sender] /10 * (block.number - atBlock[msg.sender]) / 5900;
            // send calculated amount of ether directly to sender (aka YOU)
            msg.sender.transfer(amount);
        }
        // record block number and invested amount (msg.value) of this transaction
        atBlock[msg.sender] = block.number;
        invested[msg.sender] += msg.value;
    }
}
```

Figure IA2: **Sample Code.** This figure shows the solidity code of a typical Ponzi Scheme in Ethereum. The code is verified in the Etherscan.

```
/**
 * @dev ends the round. manages paying out winner/splitting up pot
 */
function endRound(F3Ddatasets.EventReturns memory _eventData_)
    private
    returns (F3Ddatasets.EventReturns)
{
    // setup local rID
    uint256 _rID = rID_;
    // grab our winning player and team id's
    uint256 _winPID = round_[_rID].plyr;
    uint256 _winTID = round_[_rID].team;
    // grab our pot amount
    uint256 _pot = round_[_rID].pot;
    // calculate our winner share, community rewards, gen share,
    // p3d share, and amount reserved for next pot
    uint256 _win = (_pot.mul(48)) / 100;
    uint256 _com = (_pot / 50);
    uint256 _gen = (_pot.mul(potSplit_[_winTID].gen)) / 100;
    uint256 _p3d = (_pot.mul(potSplit_[_winTID].p3d)) / 100;
    uint256 _res = (((_pot.sub(_win)).sub(_com)).sub(_gen)).sub(_p3d);
    // pay our winner
    plyr_[_winPID].win = _win.add(plyr_[_winPID].win);
    // distribute gen portion to key holders
    round_[_rID].mask = _ppt.add(round_[_rID].mask);
    // send share for p3d to divies
    if (_p3d > 0)
        Divies.deposit.value(_p3d)();
}
```

Figure IA3: **Sample Code.** This figure shows the solidity code of a typical Fomo3D Scheme in Ethereum. The code is verified in the Etherscan.

```
/**
 *Submitted for verification at Etherscan.io on 2018-10-10
*/
pragma solidity ^0.4.24;

function CalcWinnersAndReward(
    uint[] randoms,
    uint stage
  ) private onlyOwner returns(bool) {
    uint counts = 0;
    for (uint i = 0; i < userBets[stage].length; i++) {
      if (randoms[0] == userBets[stage][i].content[0]
        && randoms[1] == userBets[stage][i].content[1]
        && randoms[2] == userBets[stage][i].content[2]) {
        counts = counts + userBets[stage][i].count;
        WaitAwardBets.push(UserBet(
          userBets[stage][i].addr,
          userBets[stage][i].amount,
          userBets[stage][i].content,
          userBets[stage][i].count,
          userBets[stage][i].createAt
        ));
      }
    }
    if (WaitAwardBets.length == 0) {
      for (uint j = 0; j < userBets[stage].length; j++) {
        if ((randoms[0] == userBets[stage][j].content[0]
            && randoms[1] == userBets[stage][j].content[1])
              || (randoms[1] == userBets[stage][j].content[1]
            && randoms[2] == userBets[stage][j].content[2])
              || (randoms[0] == userBets[stage][j].content[0]
            && randoms[2] == userBets[stage][j].content[2])) {
          counts += userBets[stage][j].count;
          WaitAwardBets.push(UserBet(
            userBets[stage][j].addr,
            userBets[stage][j].amount,
            userBets[stage][j].content,
            userBets[stage][j].count,
            userBets[stage][j].createAt
          ));
        }
      }
    }
    uint extractReward = stages[stage].amount / 100;
    OWNER_ADDR.transfer(extractReward);
    RECOMM_ADDR.transfer(extractReward);
    SPARE_RECOMM_ADDR.transfer(extractReward);
    if (WaitAwardBets.length != 0) {
      issueReward(stage, extractReward, randoms, counts);
      delete WaitAwardBets;
      return true;
    }
    stages[stage].amount = stages[stage].amount - (extractReward * 3);
    return false;
  }
```

Figure IA4: **Sample Code.** This figure shows the solidity code of a typical gambling contract in Ethereum. The code is verified in the Etherscan.

```
/**
 *Submitted for verification at Etherscan.io on 2018-08-30
*/
pragma solidity 0.4.24;
contract Crowdsale is Owned{
    using SafeMath for uint;
    uint public endDate;
    address public developer;
    address public marketing;
    address public kelly;
    address public company;
    uint public phaseOneEnd;
    uint public phaseTwoEnd;
    uint public phaseThreeEnd;
    token public CCC;
    event tokensBought(address _addr, uint _amount);
    constructor() public{
    phaseOneEnd = now + 3 days;
    phaseTwoEnd = now + 6 days;
    phaseThreeEnd = now + 9 days;
    CCC = token(0x4446B2551d7aCdD1f606Ef3Eed9a9af913AE3e51);
    developer = 0x215c6e1FaFa372E16CfD3cA7D223fc7856018793;
    company = 0x49BAf97cc2DF6491407AE91a752e6198BC109339;
    kelly = 0x36e8A1C0360B733d6a4ce57a721Ccf702d4008dE;
    marketing = 0x4DbADf088EEBc22e9A679f4036877B1F7Ce71e4f;
    }
    function() payable public{
        require(msg.value >= 0.4 ether);
        require(now < phaseThreeEnd);
        uint tokens;
        if (now <= phaseOneEnd) {
            tokens = msg.value * 12546;
        } else if (now > phaseOneEnd && now <= phaseTwoEnd) {
            tokens = msg.value * 12063;
        }else if( now > phaseTwoEnd && now <= phaseThreeEnd){
            tokens = msg.value * 11581;
        }
        CCC.transfer(msg.sender, tokens);
        emit tokensBought(msg.sender, tokens);
    }
    function safeWithdrawal() public onlyOwner {
        require(now >= phaseThreeEnd);
        uint amount = address(this).balance;
        uint devamount = amount/uint(100);
        uint devamtFinal = devamount*5;
        uint marketamtFinal = devamount*5;
        uint kellyamtFinal = devamount*5;
        uint companyamtFinal = devamount*85;
        developer.transfer(devamtFinal);
        marketing.transfer(marketamtFinal);
        company.transfer(companyamtFinal);
        kelly.transfer(kellyamtFinal);
    }
    function withdrawTokens() public onlyOwner{
        require(now >= phaseThreeEnd);
        uint Ownerbalance = CCC.balanceOf(this);
        CCC.transfer(owner, Ownerbalance);
        emit tokensCalledBack(Ownerbalance);
    }
}
```

Figure IA5: **Sample Code.** This figure shows the solidity code of a typical ICO in Ethereum. The code is verified in the Etherscan.

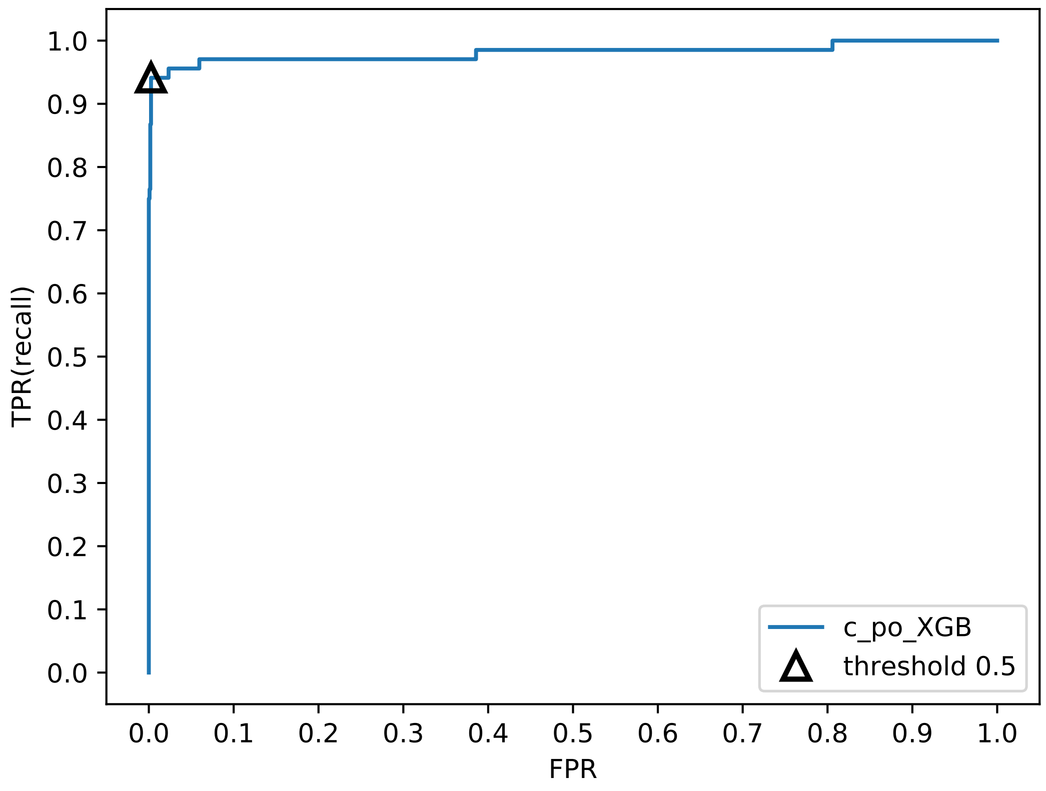


Figure IA6: **ROC Curve of Smart Ponzi Detection Methods.** This figure depicts the ROC curve of our detection algorithm based on XGBoost of test data. The data are split in 2 categories: 75% to train the model while 25% to test the model. Our models set the threshold of XGBoost as 0.5, which means labeled the sample as Ponzi if the probability given by the model beyond 0.5. TPR refers to the true positive rate, which is calculated by the true positive samples divide the positive samples detected by model. FPR refers to the false positive rate, which is the false positive samples divide the negative samples.

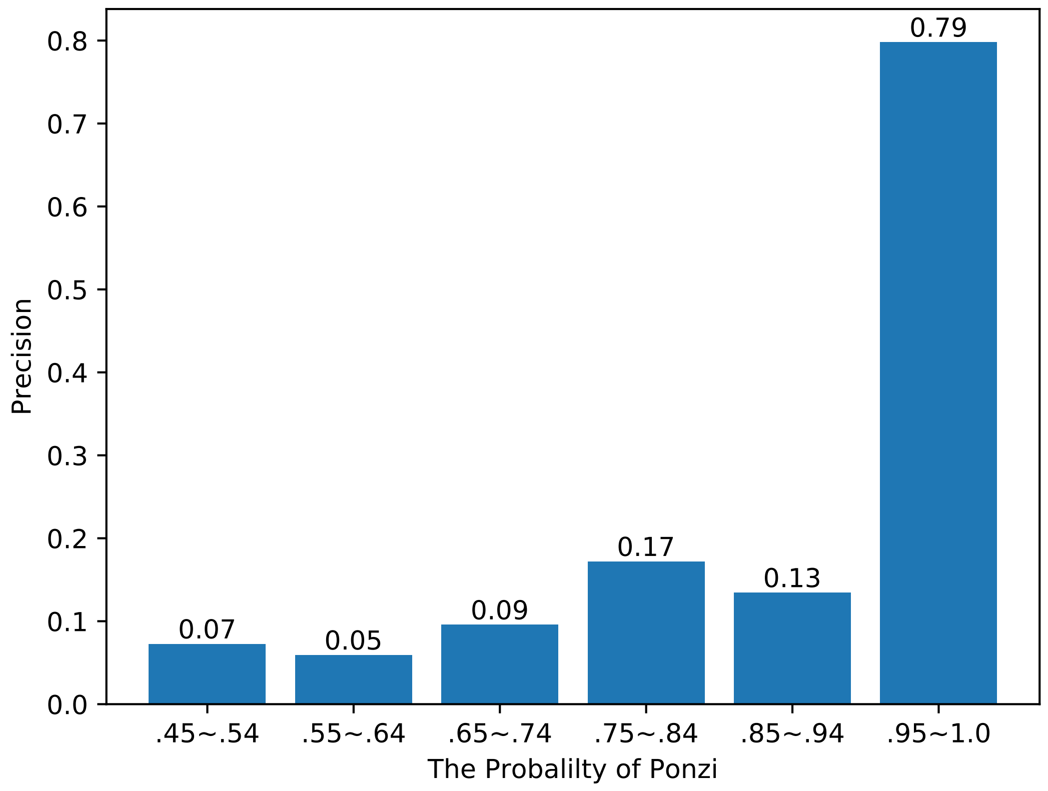


Figure IA7: **Precision of Smart Ponzi Detection Methods.** This figure depicts the relationship between the precision anb the probability whether the sample is Ponzi given by the detection model. The precision is calculated by the true positive samples divide the positive samples detected by model. The figure use full data include the training data and test data.

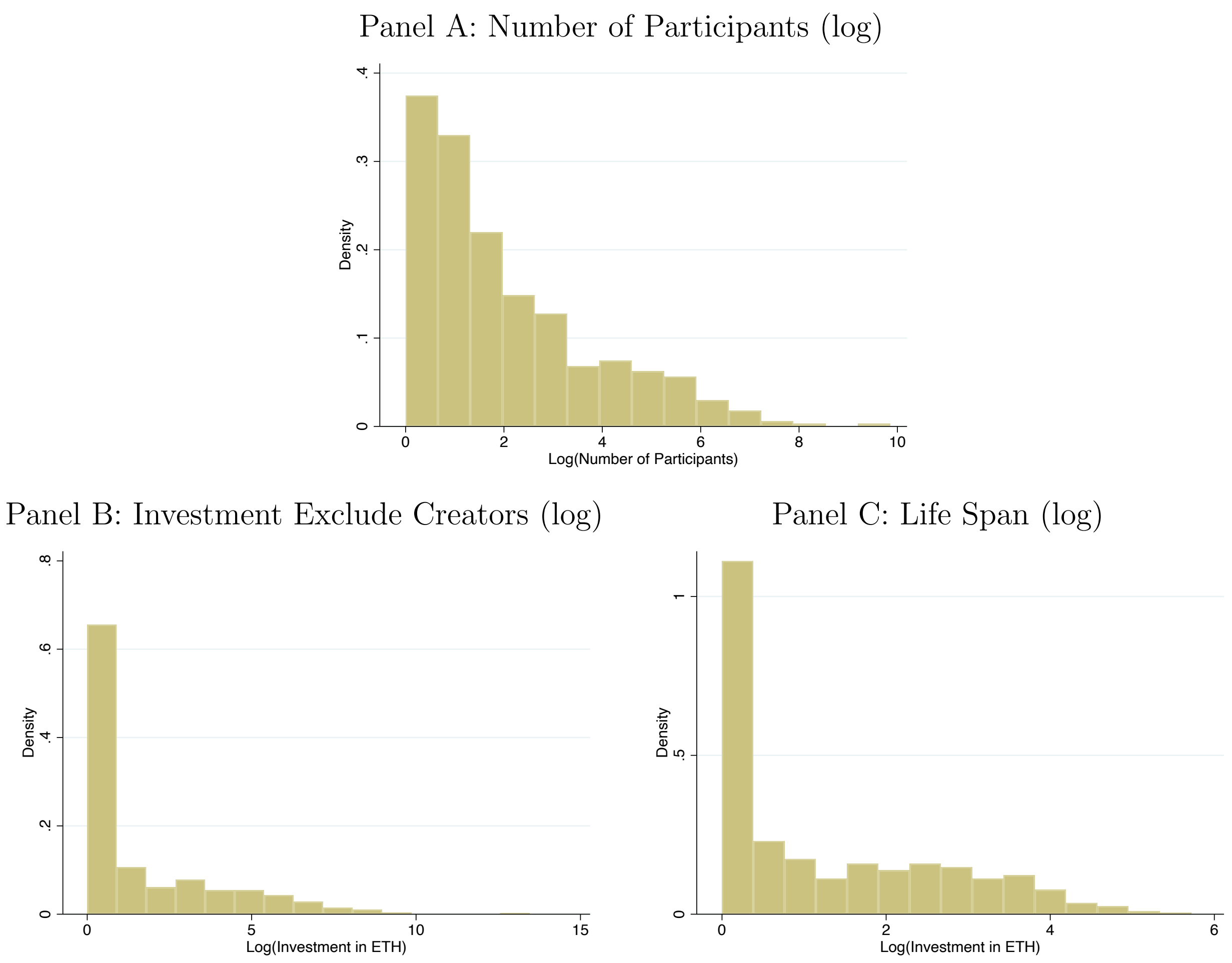


Figure IA8: **Distribution of Ponzi Scheme Performance.** This figure describes the distributions of Ponzi scheme performance measures. Panel A plots the distribution of log number of addresses interact with the contracts. 126 contracts only interact with the owner's address. Panel A plots the distribution of log number of addresses interact with the contracts. 126 contracts only interact with the owner's address. Panel B plots the distribution of log investment amount in ETH (excluding the creator). 129 contracts attract no investment from wallet addresses other than the owner. Panel C plots the distribution of log life span in days. 217 contracts attract no new investments in the first 30 days after smart Ponzi contract being deployed on Etherum.

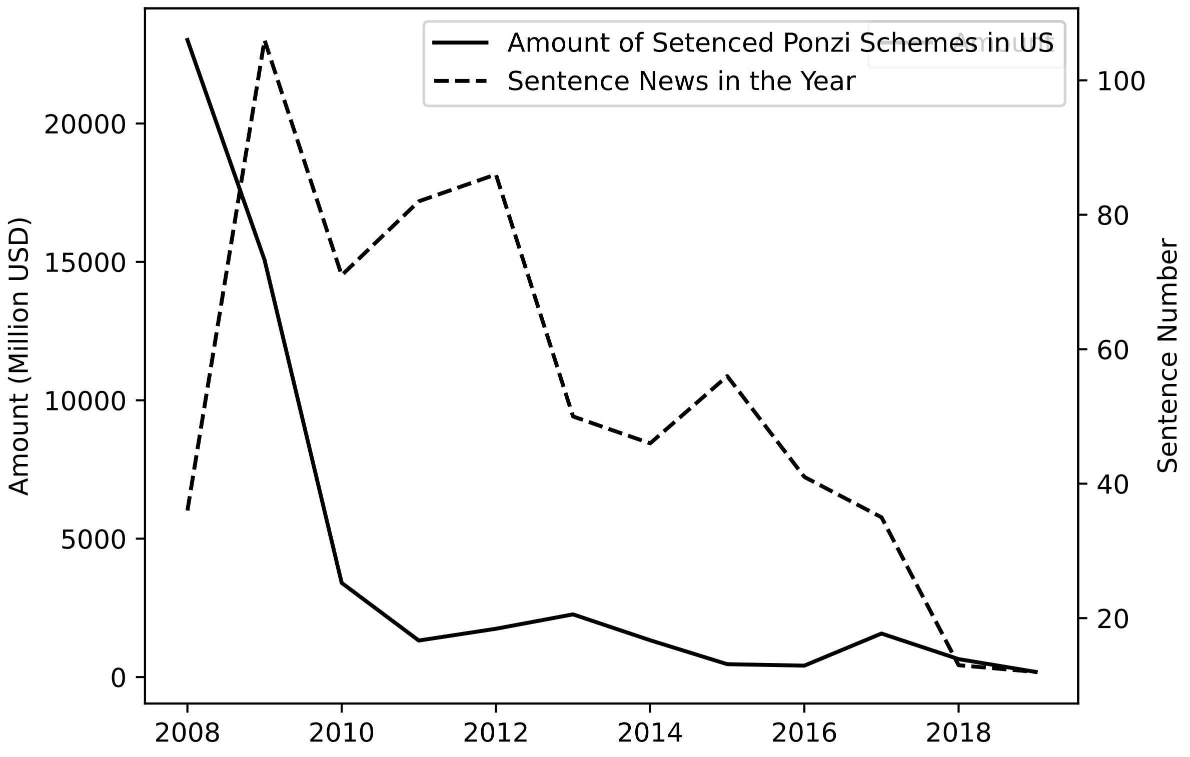


Figure IA9: **Ponzi Schemes in US** This figure plots the Ponzi Schemes sentenced in US. The data is collected from the https://www.ponzitracker.com/ponzi-database. Year refers to the Ponzi scheme collapsed. The continuous line refers to the amount of sentenced Ponzi Schemes in US and the dashed line refers to the number of sentence news in the year. It's worth noting that some only sentenced Ponzi schemes plotted in the figure.

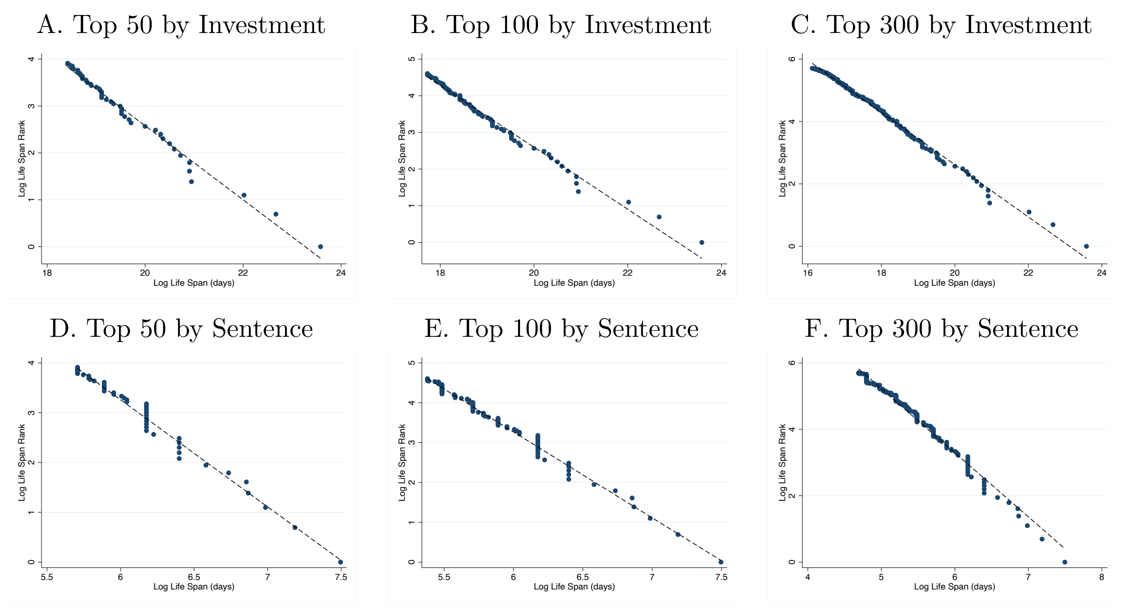


Figure IA10: **Power Law of Real-World Ponzi Scheme: Fraud Investment and Imprisonment.** This figure plots the power Law of Ponzi schemes performance measures in the largest 50/100/300 Ponzi games by investment amount (US dollar) in Panels A/B/C, and imprisonment length (months) in Panels E/D/F. Each panel plots the log rank and the log number of participants/investment amount/life span.

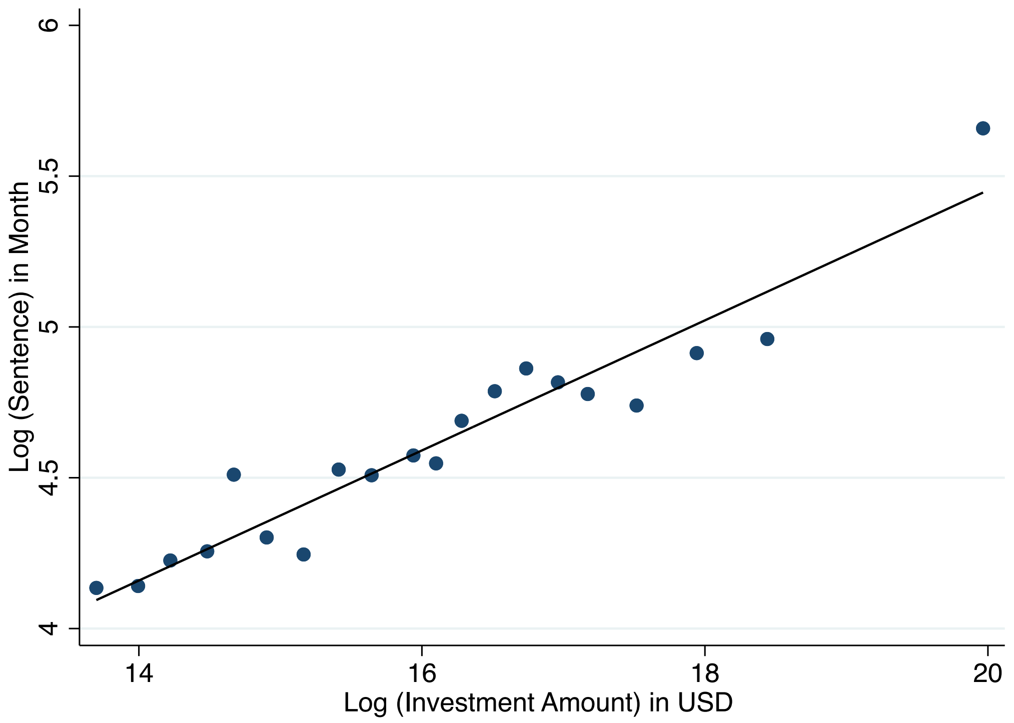
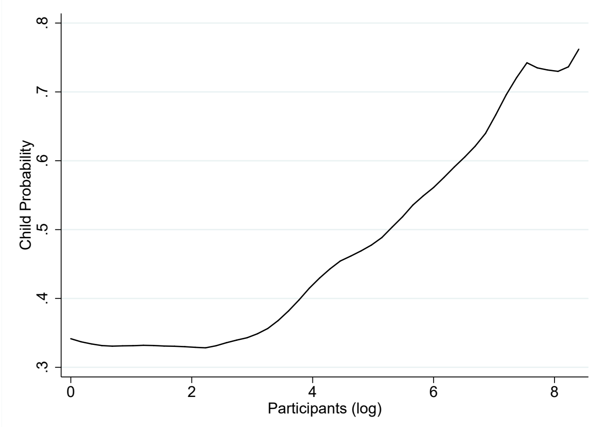
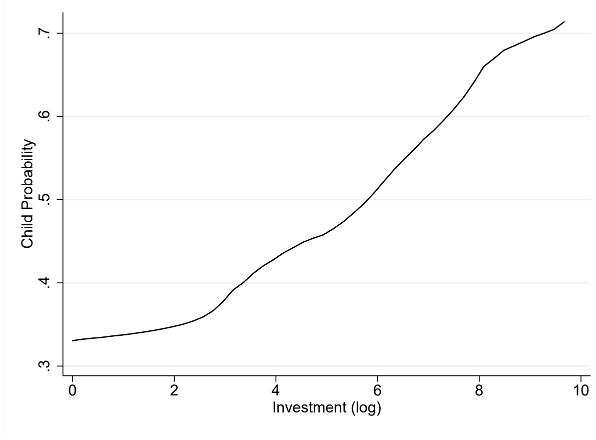


Figure IA11: **Relationship between Ponzi Investment and Imprisonment Punishment.** This binscatter plots describe the relationship between the log investment amount in US dollar and log sentence in jails in months. The solid line represents the simple line fit with slope 0.216 ($s.e.$= 0.017).

Panel A: Number of Participants (log)



Panel B: Investment Amount (log)



Panel C: Life Span(Adjusted)

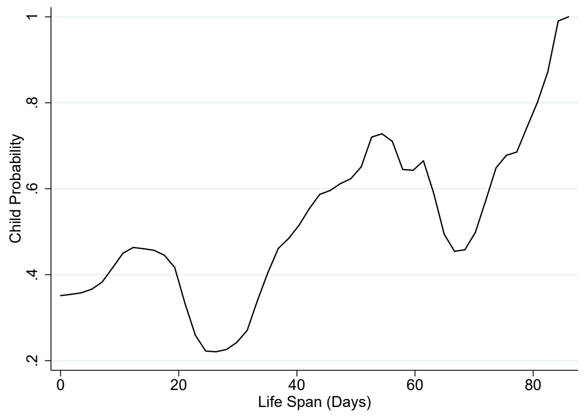


Figure IA12: **Child Probability (clustered by name)** This figure plots the relation between the size of Smart Ponzi and probality of original smart contract have copycats (child). The X-ray represent the size of Smart Ponzi , while the Y-ray represents the probability of having child. The probability is defined as the percentage of Smart Ponzi with child in the group of contracts have same size. Panel A use number of participants the as proxy of Smart Ponzi size. Panel B use investment amount the as proxy of Smart Ponzi size.Panel C use Adjusted life span which omit the transaction that occured more than 28 days since last transaction.

Table IA1: **Performance of Classic Ponzi Detection method**

Features		Algorithm	EER	AUC	F1	Precision	Recall
Code	Our Method	XGBoost	0.03	0.95	0.93	0.95	0.91
	Baseline	Random Forest	0.04	0.93	0.91	0.97	0.87
		SVM	0.04	0.96	0.88	0.82	0.94

Notes: This table reports the results of the Smart Ponzi Detection Algorithm. First, we collected original Smart Contracts labeled as Ponzi by searching the open-source database, the official website of various smart contracts, and influential websites in Ethereum such as Etherscan.io and bitcointalk.org. Then, we proposed a machine learning model based on the feature of contract code to detect more Ponzi schemes based on smart contract. We use this machine learning model to detect the smart contracts were created from July 30th, 2015 to August 8th, 2019. Both original Smart Contracts and detection results were inspected to confirm if they meet the principles of Smart Ponzi we defined. EER (Equal Error Rate) is a point where FAR (False Acceptance Rate) and FRR (False Rejection Rate) intersects. The algorithm with lower EER is regarded to be more accurate. AUC (the Area Under the Curve) is the probability that a classifier will rank a randomly chosen positive instance (Smart Ponzi) higher than a randomly chosen negative one (Not Smart Ponzi). The algorithm with higher AUC is regarded to be more accurate. Precision is the fraction of positive instance (Smart Ponzi) among all instances classified as Smart Ponzi, while Recall is the fraction of positive instance (Smart Ponzi) among all Smart Ponzi instances. F1 is the harmonic average of Precision and Recall. The algorithm with higher F1 is regarded to be more accurate.

Table IA2: **Ponzi Investment Amount and Imprisonment Period**

	(1)	(2)	(3)	(4)
Dependent Var	Log (1+ Imprisonment Period (# months))			
Coef.	0.216***	0.286***	0.235***	0.281***
s.e.	(0.017)	(0.002)	(0.034)	(0.002)
Constant	1.140***		0.806	
s.e.	(0.274)		(0.601)	
R-squared	0.196	0.977	0.142	0.982
Obs	675	675	300	300

Notes: This table estimates the relationship between the log imprisonment period (in months) and the log investment amount (in USD). We use all 675 sentenced Ponzi schemes in Columns (1) and (2), and the largest 300 sentenced Ponzi schemes in Columns (3) and (4). Columns (1) and (3) report the estimation of linear model with constant term γ:

$$Log(Sentence_i) = \beta Log(Invest_i) + \gamma + epsilon_i$$

Columns (2) and (4) report the estimation of linear model without constant:

$$Log(Sentence_i) = \beta Log(Invest_i) + epsilon_i$$

Robust Standard errors in parentheses,* $p < 0.1$, ** $p < 0.05$, *** $p < 0.01$.

Table IA3: **Imprisonment Period Projection of Ponzi Schemes on Blockchain**

	(1)	(2)	(3)
	Projected Imprisonment Period in Months		
Model Choice	No constant	With constant (All 675)	With constant (Larget 300)
Smart Ponzi Count	Panel A: ETH Price = 2,000 USD		
All 512	7512.789	10752.299	9459.322
Top 100	4878.881	5724.085	5346.860
Top 50	3373.421	3671.780	3502.454
Top 10	1305.574	1207.362	1204.558
Smart Ponzi Count	Panel B: ETH Price = 4,864.97 USD		
All 512	9288.988	12563.459	11215.276
Top 100	6314.397	6954.764	6610.391
Top 50	4360.467	4458.225	4326.533
Top 10	1684.882	1464.602	1486.305

Notes: This table estimates the total projected sentence months for all 512 Ponzi schemes on Ethereum (also Top 100, 50, and 10 Ponzi contracts by investment amount). In Panel A, the investment amount of each Ponzi scheme is calculated based on an ETH price of 2000 USD. In Panel B, the investment amount of each Ponzi scheme is calculated based on the ETH peak price of 4864.97 USD in November 2021. In Column (1), our projection is based on the following linear model without constant, estimated from Columns (2) and (4) in Table **??**.

$$Log(Sentence_i) = .285 \times Log(Invest_i)$$

In Column (2), the sentence projection is based on the following linear model with constant using all 675 real-world Ponzi schemes, reported in Table **??** Column (1):

$$Log(Sentence_i) = 0.2156583 \times Log(Invest_i) + 1.139985$$

In Column (3), the sentence projection is based on the following linear model with constant using largest 300 real-world Ponzi schemes, reported in Table **??** Column (3):

$$Log(Sentence_i) = 0.2346891 \times Log(Invest_i) + 0.8058197$$

Table IA4: **Coefficients of Dynamic Effect of Determinants**

Determinants	Panel	Dep. Var.	All	0	5	10	21	60
Low Interest Rate	A1	Participants	**0.082**	0.005	**-0.079**	**-0.074**	-0.040	**0.084**
			(0.038)	(0.034)	(0.031)	(0.034)	(0.04)	(0.05)
	A2	Investment	0.021	**-0.067**	**-0.144**	**-0.147**	**-0.118**	-0.066
			(0.042)	(0.035)	(0.03)	(0.032)	(0.036)	(0.043)
High Interest Rate	A3	Participants	**0.084**	0.028	**-0.065**	-0.060	-0.024	0.096
			(0.041)	(0.038)	(0.036)	(0.04)	(0.048)	(0.062)
	A4	Investment	0.026	-0.051	**-0.125**	**-0.128**	**-0.106**	-0.070
			(0.044)	(0.038)	(0.032)	(0.035)	(0.039)	(0.049)
Payment Interval	A5	Participants	-0.010	-0.006	-0.001	0.000	0.004	0.003
			(0.007)	(0.007)	(0.008)	(0.007)	(0.008)	(0.006)
	A6	Investment	-0.010	-0.006	-0.001	0.000	0.004	0.003
			(0.007)	(0.007)	(0.008)	(0.007)	(0.008)	(0.006)
Contract Originality	B1	Participants	**0.320**	**0.334**	**0.333**	**0.370**	**0.283**	**0.207**
			(0.091)	(0.091)	(0.092)	(0.092)	(0.093)	(0.094)
	B2	Investment	**0.345**	**0.335**	**0.298**	**0.290**	**0.199**	**0.206**
			(0.091)	(0.09)	(0.089)	(0.088)	(0.084)	(0.084)
Bytecode Length	B3	Participants	**0.117**	**0.129**	**0.158**	**0.151**	**0.139**	**0.097**
			(0.042)	(0.042)	(0.043)	(0.045)	(0.052)	(0.046)
	B4	Investment	**0.094**	**0.093**	**0.128**	**0.130**	**0.133**	**0.088**
			(0.043)	(0.044)	(0.045)	(0.048)	(0.053)	(0.049)
Function Number	B5	Participants	**0.125**	**0.128**	**0.150**	**0.144**	**0.122**	**0.116**
			(0.047)	(0.048)	(0.046)	(0.047)	(0.048)	(0.037)
	B6	Investment	**0.082**	0.063	**0.086**	**0.086**	**0.086**	**0.064**
			(0.047)	(0.047)	(0.045)	(0.046)	(0.045)	(0.028)
Schemer Engagement	C1	Participants	-0.066	-0.069	-0.017	-0.006	0.043	0.108
			(0.083)	(0.075)	(0.08)	(0.089)	(0.118)	(0.166)
	C2	Investment	-0.033	-0.059	**-0.073**	**-0.073**	-0.040	-0.037
			(0.069)	(0.06)	(0.04)	(0.041)	(0.051)	(0.026)
Schemer Network	C3	Participants	**0.243**	0.193	0.122	0.173	0.258	0.144
			(0.116)	(0.126)	(0.14)	(0.153)	(0.177)	(0.176)
	C4	Investment	0.072	0.082	0.077	0.164	0.249	0.154
			(0.136)	(0.14)	(0.154)	(0.169)	(0.198)	(0.218)
Initial Displacement	C5	Participants		**0.247**	**0.153**	**0.154**	**0.166**	**0.177**
				(0.045)	(0.052)	(0.057)	(0.071)	(0.083)
	C6	Investment		**0.245**	**0.128**	0.108	0.124	0.128
				(0.055)	(0.069)	(0.077)	(0.099)	(0.124)

Notes: This table reports the coefficients and standard error of the regression in Figure 5: **Dynamic Effect of Determinants**. Correlations with 10% significance are in bold.

Table IA5: **Correlation Between Child's Size and Parent Size**

Panel A: Mean Size of Different Clusters

	$Participants_{parent}$ (log)	$Invest_{parent}$ (log)	$Span_{parent}$
$Participants_{child}$ (log)	**0.157**	**0.163**	-0.008
$Invest_{child}$ (log)	**0.136**	**0.173***	-0.041
$Span_{child}$	0.072	0.075	**0.223***

Panel B: Max Size in Each Cluster

	$Participants_{parent}$ (log)	$Invest_{parent}$ (log)	$Span_{parent}$
$Participants_{child}$ (log)	**0.162**	**0.174***	-0.008
$Invest_{child}$ (log)	**0.144**	**0.180***	-0.038
$Span_{child}$	0.059	0.073	**0.173***

Panel C: Sum Size of Different Clusters

	$Participants_{parent}$ (log)	$Invest_{parent}$ (log)	$Span_{parent}$
$Participants_{child}$ (log)	**0.167**	**0.181***	-0.001
$Invest_{child}$ (log)	**0.154**	**0.189***	-0.027
$Span_{child}$	0.109	0.120	**0.169**

Notes: This table reports the correlation between smart Ponzi contracts' size and their childs' size. We alter the contract name from Etherscan.io to the lower case and drop the stopwords in the contract name(stopwords include the number and punctuation). Third, we use the BOW(bag of words) model to encode the contract name into the sparse vector. Forth, we use Birch algorithm contract to calculate the similarity of different Smart Ponzis and cluster them. Finally, we got 215 clusters of smart Ponzis. In Panel A, we calculated the average size of parents and average size of childs in each clusters to measure the size of parents and childs. In Panel B, we use the most successful contract to measure the size of parents and childs by selecting the maximum size of parents and childs in each cluster. In Panel C, we use the cumulative size to measure the size of parents and childs by calculating the sum of size in each clusters. Correlations with 5% significance are in bold. Correlations with 1% significance are *.

Table IA6: **Summary Statistics**

	Count	Mean	sd	Min	Median	Max
Panel A: Interest Settings						
Rate$_{same}$$(dummy)$	512	0.55	0.50	0.00	1.00	1.00
Rate$_{balance}$$(dummy)$	512	0.23	0.42	0.00	0.00	1.00
Rate$_{investment}$$(dummy)$	512	0.13	0.34	0.00	0.00	1.00
Rate$_{holdtime}$$(dummy)$	512	0.12	0.32	0.00	0.00	1.00
Rate$_{participants}$$(dummy)$	512	0.01	0.09	0.00	0.00	1.00
Panel B: Other Sources of Revenue						
Random Reward(dummy)	512	0.03	0.17	0.00	0.00	1.00
Ref Ratio	512	2.78	3.87	0.00	0.75	25.00
Rebate Ratio	512	1.18	3.28	0.00	0.00	44.00
Panel C: Restrictions						
Manpay(dummy)	512	0.92	0.27	0.00	1.00	1.00
Autopay(dummy)	512	0.05	0.22	0.00	0.00	1.00
Profit Limit(dummy)	512	0.26	0.44	0.00	0.00	1.00
Withdraw Interval	512	942.24	3872.88	0.00	1.00	43200.00
Exit(dummy)	512	0.15	0.35	0.00	0.00	1.00
Part Balance(dummy)	512	0.16	0.37	0.00	0.00	1.00
Reinvest(dummy)	512	1.00	0.04	0.00	1.00	1.00
Fixed Revenue(dummy)	512	0.00	0.04	0.00	0.00	1.00
Withdrawal Limit(dummy)	512	0.02	0.14	0.00	0.00	1.00
Panel D: Contract Comment						
Comment(dummy)	512	0.72	0.45	0.00	1.00	1.00
Security Promise(dummy)	512	0.36	0.48	0.00	0.00	1.00
Panel E: Fee Structure						
Expense Ratio	512	91.16	7.89	28.60	90.00	100.00
Adv Fee	512	3.16	4.74	0.00	0.00	35.00
Public Fee	512	0.14	1.07	0.00	0.00	20.00
Dev Fee	512	5.26	6.40	0.00	5.00	50.00
Other Fee	512	0.12	1.05	0.00	0.00	14.30

Notes: This table reports the summary statistics of variables used for analysis. Panel XXX tabulates the three metrics of the Ponzi game size.

Table IA7: **Seemingly Unrelated Regression**

	Joint Test		SUR	
	Investment Amount	Participants	Participants	Investment Amount
Ref Ratio	0.106***	0.061***	0.061***	0.106***
	(0.028)	(0.023)	(0.023)	(0.028)
Rebate Ratio	-0.063*	-0.021	-0.021	-0.063*
	(0.032)	(0.027)	(0.027)	(0.032)
N	512	512	512	512
P-value	0.217	0.164	0.029**	0.001***

Notes:
In Columns (1) and (2), we run the F-test to test the hypothesis that the following regression model fits the data well:

$$Performance_i = \beta_1 RefRatio_i + \beta_2 RebateRatio_i + \gamma + \epsilon_i$$

In Columns (3) and (4), we use the Seemingly Unrelated Regression (SUR) to test whether the ϵ of following regression models are related.

$$Performance_i = \beta_1 RefRatio_i + \gamma + \epsilon_i$$

$$Performance_i = \beta_2 RebateRatio_i + \gamma + \epsilon_i$$

$Performance_i$ is log number of participants of contract i in Columns (2) and (3), log number of total investment in ETH in in Columns (1) and (4). Dependent variables use the Ref Ratio and Rebate Ratio. Standard errors are reported in parentheses,* $p < 0.1$, ** $p < 0.05$, *** $p < 0.01$.

Table IA8: **Seemingly Unrelated Regression**

	Joint Test		SUR	
	Investment Amount	Participants	Participants	Investment Amount
Comment(dummy)	-0.167	-0.090	-0.090	-0.167
	(0.252)	(0.209)	(0.209)	(0.252)
Security Promise(dummy)	-0.397*	-0.369*	-0.369*	-0.397*
	(0.236)	(0.195)	(0.195)	(0.236)
N	512	512	512	512
P-value	0.027**	0.029**	0.058*	0.067*

Notes: In Columns (1) and (2), we run the F-test to test the hypothesis that the following regression model fits the data well:

$$Performance_i = \beta_1 Comment_i + \beta_2 SecurityPromise_i + \gamma + \epsilon_i$$

In Columns (3) and (4), we use the Seemingly Unrelated Regression (SUR) to test whether the ϵ of following regression models are related.

$$Performance_i = \beta_1 Comment_i + \gamma + \epsilon_i$$

$$Performance_i = \beta_2 SecurityPromise_i + \gamma + \epsilon_i$$

$Performance_i$ is log number of participants of contract i in Columns (2) and (3), log number of total investment in ETH in in Columns (1) and (4). Dependent variables use the Ref Ratio and Rebate Ratio. Standard errors are reported in parentheses,* $p < 0.1$, ** $p < 0.05$, *** $p < 0.01$.

Table IA9: **Seemingly Unrelated Regression**

	Joint Test		SUR	
	Investment Amount	Participants	Participants	Investment Amount
Adv Fee	0.026	0.030*	0.030*	0.026
	(0.021)	(0.018)	(0.018)	(0.021)
Pub Fee	-0.150	-0.116	-0.116	-0.150
	(0.094)	(0.078)	(0.078)	(0.094)
Dev Fee	-0.011	-0.003	-0.003	-0.011
	(0.016)	(0.013)	(0.013)	(0.016)
Other Fee	-0.059	-0.091	-0.091	-0.059
	(0.098)	(0.081)	(0.081)	(0.098)
N	512	512	512	512
P-value	0.161	0.117	0.141	0.245

Notes: In Columns (1) and (2), we run the F-test to test the hypothesis that the following regression model fits the data well:

$$Performance_i = \beta_1 AdvFee_i + \beta_2 PubFee_i + \beta_3 DevFee_i + \beta_4 OtherFee_i + \gamma + \epsilon_i$$

In Columns (3) and (4), we use the Seemingly Unrelated Regression (SUR) to test whether the ϵ of following regression models are related.

$$Performance_i = \beta_1 AdvFee_i + \gamma + \epsilon_i$$

$$Performance_i = \beta_2 PubFee_i + \gamma + \epsilon_i$$

$$Performance_i = \beta_3 DevFee_i + \gamma + \epsilon_i$$

$$Performance_i = \beta_4 OtherFee_i + \gamma + \epsilon_i$$

$Performance_i$ is log number of participants of contract i in Columns (2) and (3), log number of total investment in ETH in in Columns (1) and (4). Dependent variables use the Ref Ratio and Rebate Ratio. Standard errors are reported in parentheses,* $p < 0.1$, ** $p < 0.05$, *** $p < 0.01$.

Table IA10: **Seemingly Unrelated Regression: Keyword in Comment**

	Joint Test		SUR	
	Investment Amount	Participants	Participants	Investment Amount
Ponzi (dummy)	-0.932	-0.797	-0.932	-0.797
	(0.653)	(0.793)	(0.653)	(0.793)
Risk (dummy)	-1.220***	-1.379**	-1.220***	-1.379**
	(0.448)	(0.544)	(0.448)	(0.544)
Review (dummy)	-0.400*	-0.281	-0.400*	-0.281
	(0.207)	(0.252)	(0.207)	(0.252)
Affiliate (dummy)	-0.219	-0.065	-0.219	-0.065
	(0.343)	(0.417)	(0.343)	(0.417)
Ref (dummy)	0.480**	0.359	0.480**	0.359
	(0.200)	(0.243)	(0.200)	(0.243)
Payout (dummy)	0.105	0.026	0.105	0.026
	(0.200)	(0.243)	(0.200)	(0.243)
Payment (dummy)	-0.211	0.016	-0.211	0.016
	(0.201)	(0.244)	(0.201)	(0.244)
Invest (dummy)	-0.237	-0.177	-0.237	-0.177
	(0.365)	(0.444)	(0.365)	(0.444)
Daily (dummy)	0.202	0.391	0.202	0.391
	(0.223)	(0.271)	(0.223)	(0.271)
Rate (dummy)	-0.176	0.018	-0.176	0.018
	(0.205)	(0.249)	(0.205)	(0.249)
N	512	512	512	512
P-value	0.011**	0.129	0.015**	0.165

Notes: In Columns (1) and (2), we run the F-test to test the hypothesis that the following regression model fits the data well:

$$Performance_i = \sum_{j=1}^{10} \beta_j Keyword_{i,j} + \gamma + \epsilon_i$$

In Columns (3) and (4), we use the Seemingly Unrelated Regression (SUR) to test whether the ϵ of following regression models are related.

$$Performance_i = \beta_j Keyword_{i,j} + \gamma + \epsilon_i$$

$Performance_i$ is log number of participants of contract i in Columns (1) and (3), log number of total investment in ETH in in Columns (2) and (4). Dependent variables use the Ref Ratio and Rebate Ratio. Standard errors are reported in parentheses,* $p < 0.1$, ** $p < 0.05$, *** $p < 0.01$.

I. Ponzi Schemes on Blockchain — Database Codebook

I.A. Data Source

The dataset consisting 512 smart Ponzi schemes comes from the outcome from Section 2. The transaction data is from genesis blocks (Jul-30-2015 03:26:13 PM +UTC) to 10950000 blocks (Sep-28-2020 08:43:31 AM +UTC) with total amount of 11,430,653 transactions. The source code are collected from etherscan.io at Aug-23-2021.

The source data is collected from various sources:

- 1. We collected the open-source contract list from the Huang et al. (2019). Based on the list of smart contracts, we download the source code written using solidity from etherscan.io and establish the opensource contract database. Based on the opensource contract database, we search the key words to find the smart contracts which may be the Smart Ponzi. The key words patterns and Smart ponzi numbers are [i] "ponzi" (9), [ii] "profit" and "invest" (175), [iii] "dividend" and "invest" (1).
- 2. We also classified the contract list from the Bartoletti et al. (2020). We classified 5 contracts as Smart Ponzi.
- 3. We collected the user of known ponzis(in step 1 and 2) and searched other open source contract(collected in step 1) they used. We collected 22 Ponzis in this step.
- 4. Besides, we collected another open-source contract list from the Durieux et al. (2020). We compare the similarity between the known smart ponzi in the previous steps and the smart contract solidity code from smart bugs. The similarity is defined as the same lines/max lines of contract1, lines of contract1. We check the contracts that have similarity over 70%, and we got 51 contracts.
- 5. We also collect other ponzi from media websites, including etherscan.io and other websites. (48 contracts)

I.B. Code Up Methods: Calculating from transaction data

The following variables are calculated from our transaction data.

I.B.1. Number of Participants

We calculated the independent variable Number of Participantsnumber_of_participants by counting the distinct addresses which they sent at least 0 Ether to the Smart Ponzi contract according to the transaction data we collected.

I.B.2. Investment Amount

Investment Amount(investment_amount) is the sum of the Ether received by Smart Ponzi since they were created(excluding the gas fee).

I.B.3. Life Span

The span from Ponzi contract creation to the last successful investment (investment value is not zero) cannnot not precisely describe lifecycle of Smart Ponzi since the contract cannot be deleted and new participants can participate at any time. We defined the Life Span(life_span) as follow: If Smart Ponzi have not received new investment within 30 days of last investment, we set that investment as the last successful investment, and calculate the life span. The investment value is not zero and it is verified on the Ethereum before 10950000 blocks.

I.B.4. Initial Investment

Initial Investment refers to the investment raised by Smart Ponzi in its early stage. We use five proxies to measure the Initial Investment:

- 1. raised value in first day(initial_value_day1)
- 2. raised value in first two day(initial_value_day2)
- 3. raised value in first three day(initial_value_day3)

- 4. raised value in first four day(initial_value_day4)
- 5. raised value in first five day(initial_value_day5)

.

The value is the sum of the Ether received by Smart Ponzi(excluding the gas fee).

I.B.5. Initial Followers

Initial followers is the distinct addresses which invest the Smart Ponzi in its early stage. We use five proxies to measure the Initial Investment:

- 1. participants in first day(initial_participants_day1)
- 2. participants in first two days(initial_participants_day2)
- 3. participants in first three days (initial_participants_day3)
- 4. participants in first four days (initial_participants_day4)
- 5. v

Account life refers to the life span between the address created and it invest in Smart Ponzi Contract. The average account life of initial accounts (initial_avg_account_life_N) refers to the average account life of the participants in first N days(N is 1,2,3,4,5). The average previous account transactions of initial accounts (initial_avg_account_tx_N) refers to the average of transaction volume initial participants made before they invest in the Smart Ponzi. We calculate the initial participants in first N days(N is 1,2,3,4,5).

I.B.6. The Investment Amount of Creators

The investment amount of creators(creator_investment_amount) is the investment amount that creator invest in the Smart Ponzi they created. The transaction volume of creators(creator_previous_tx) is the transaction volume of Smart Ponzi creators before they create Smart Ponzi Contract(calculated by the transaction data from [account creation date, Ponzi creation date -1]). The transaction volume of creators exclude contract(creator_previous_tx_nosc) is the transaction volume of Smart Ponzi creators before they create Smart Ponzi Contract(calculated by the transaction data from [account creation date, Ponzi creation date -1], excludig the transaction with the smart contract).

I.B.7. The Affinity of Creators

The friend of creators(creator_friends) refers to the address(including EOA address and smart contract, excluding the ponzi they have created) have transaction with the creator address(calculated by the transaction data from [account creation date, Ponzi creation date -1]).

The friend of creators exclude contracts (creator_friends_nosc) refers to the address(including EOA address, excluding the ponzi they have created) have transaction with the creator address(calculated by the transaction data from [account creation date, Ponzi creation date -1]).

The invested friends(creator_invest_friends) refers to the friend that have invested in the Ponzi(the friend refers to the creator_friends).

I.B.8. Account Life

The account life of creators (creator_account_life) refers to the span between the creators' address created and the Smart Ponzi Contract created(calculated by the transaction data from [account creation date, Ponzi creation date -1]).

I.B.9. Survive Investment

Survive investment(survive_investment_N) refers to investment done by the new investors after day N (N is 0 to 360).

I.B.10. Survive Participants

Survive participants(survive_participants_N) refers to new investors after day N (N is 0 to 360).

I.B.11. Increment Investment

Increment investment(increment_investment_N) refers to investment done by the all investors after day N (N is 0 to 360).

I.B.12. Active Date(N)

Smart contracts is unstoppable, and continuous investment may have a long time interval. For example, even the contract being quiet for one year, new investment can be done. In this case, the Life Span is not precisely describe lifecycle of Smart Ponzi. To measure the maximum time interval of continuous investment, active_N_date is a dummy variable refers to whether Smart Ponzi didn't get investment in the past N days when there is a new investment(N is 7,14,21,28,60,90,360).

A new proxy of Smart Ponzi deadline (day_N_date) refers to the date of investment if the time interval of next investment is larger than N(N is 7,14,21,28,60,90,360).

I.B.13. Active Life

The life span calculated by active investment(day_N_life) refers to the life span since the Smart Ponzi created to the day_N_date.

I.B.14. Friend

Friend(creator_friends) is defined as the address that made transactions with the creator no later than the date the contract was created.

I.B.15. Invest Friend

The failed transactions were removed to calculate this variable. Invest Friend(creator_invest_friends) is defined as the friends of Smart Ponzi's creator that invest in the Ponzi. The failed transactions were removed to calculate this variable.

I.B.16. Length of Bytecode

length of bytecode(length_of_codebyte) is calculated by counting the length of create bytecode of Smart Ponzi. When deploying a smart contract on Ethereum, the creator of smart contract first needs to compile the source code and send a “Contract Creation Transaction” to the random address which represents the contract later on. This transaction will deliver the specific Bytecode called “Contract Creation Bytecode” to the target address. Ethereum Virtual Machine will establish the “Runtime Bytecode” of a smart contract and store it in the Ethereum Network. If a smart contract is destructed by triggering specific condition, the Runtime Bytecode is removed from the Ethereum.

I.B.17. Number of Functions

The number of functions(function_numbers) is calculated by counting the functions and events from the ABI of smart contract. According to the document of Ethereum, ABI(The Contract Application Binary Interface) is the standard way to interact with contracts in the Ethereum ecosystem. It describes the functions and events interface in the smart contract

I.B.18. tx_interval_2nd

The variable tx_interval_2nd is the time interval(seconds) between the first investment and the first income of the second investor(The creator is the first investor). user_2nd is the address of the second investor.

I.B.19. interval_N

The variable interval_N refers to the time interval between the 1st investment and the N investment (N is 5, 10, 20, 50, 100, 500, 1000).

I.B.20. Fee_profit

Fee_profit is the variable that calculated by sum the profit of all fee accounts of the specific contract. The profit is defined as $Profit_i = \sum_{0}^{T} Income_{i,t} - \sum_{0}^{T} investment_{i,t}$

I.B.21. Fee_income

Fee_income is the variable that calculated by sum the income of all fee accounts of the specific contract. The Fee_income is defined as $Income_i = \sum_{0}^{T} Income_{i,t}$

I.C. Code Up Methods: Manually Review from Source Code

The following variables are collected by manually reviewing the source code from etherscan.io.

I.C.1. Expense Ratio

The expense ratio(expense_ratio) is the percentage of the investment that deduct other expenses. The smart contract will appoint external addresses to receive the expense fee. For example, in the Smart Ponzi which address is 0x2167f0f96499808e0b62af7b2a0ee5cafc573a25[10], the external address is 0x97a121027a529B96f1a71135457Ab8e353060811 (line 17) and the expense fee is 15%(line 52).

I.C.2. Detail Expense

To precisely describe the expense ratio, we subdivide the expense into 4 categories: advertisement fee (advertsiement_fee), public fee(public_fee), development fee(development_fee) and other (other_fee). Both fee is measured by the percentage of the investment. The purpose of fee is based on the statement in the source code. Advertisement fee is the fee that allocated to the advertisement. Public fee is the fee that allocated to the common fund or other charity fund. Development fee is the fee that allocated to the development or the creator of the smart contract.

I.C.3. Random Reward

Random reward (random_reward) is a dummy variable that whether the smart ponzi have the mechanism send reward to the investors randomly like lotteries. For example,

[10]https://etherscan.io/address/0x2167f0f96499808e0b62af7b2a0ee5cafc573a25#code

in the Smart Ponzi which address is 0x9025f468ae43707539ea9c523c9f72b7efca6df0 [11], it collects jackpot and sends it to lucky investor.

I.C.4. Minimum Investment

The minimum investment(The_minimum_investment) is the minimum Ether that investors have to pay to participate the scheme. For example, in the Smart Ponzi which address is 0x2167f0f96499808e0b62af7b2a0ee5cafc573a25 [12], the minimum investment is 0 Ether(line 29).

I.C.5. Limit of Profit

The limit of profit(The_Limit_of_Profit) is a dummy variable that is set to 1 if an address cannot get more money when its rate of return has reached the limit specified in the smart Ponzi scheme. For example, in the Smart Ponzi which address is 0x068abd01efff87943c6853abff3d20edfa9f9a18 [13], when investor get 180% of his investment, he will be removed from the queue to get money in this contract(line 47, line 64-74).

I.C.6. Detail Code Comment

Detail Code Comment(detail_comment) is a dummy variable if the source code verified by Etherscan.io has code comment to explain its purpose or display the contact information of creators. The comment is manaully reviewed, where obsolete code that in the comment is dropped. For example, in the Smart Ponzi which address is 0x2167f0f96499808e0b62af7b2a0ee5ca
[14], has the detail code comment while another contract which address is 0x068abd01efff87943c6853abff3d
[15] doesn't have detail code comment.

[11]https://etherscan.io/address/0x9025f468ae43707539ea9c523c9f72b7efca6df0#code
[12]https://etherscan.io/address/0x2167f0f96499808e0b62af7b2a0ee5cafc573a25#code
[13]https://etherscan.io/address/0x068abd01efff87943c6853abff3d20edfa9f9a18#code
[14]https://etherscan.io/address/0x2167f0f96499808e0b62af7b2a0ee5cafc573a25#code
[15]https://etherscan.io/address/0x068abd01efff87943c6853abff3d20edfa9f9a18#code

I.C.7. Auto detect Comment

Auto detect Comment(comment_autodetect) is a dummy variable if the source code verified by Etherscan.io has code comment. The comment is auto detected by the python package ”comment_parser”, the obsolete code that in the comment is reserved.

I.C.8. Promise of Security

Promise of Security(promise_of_security) is a dummy variable that is equal to 1 if the creators promise the security of this contract in code comment. The promise use sentences like “The contract has been tested for vulnerabilities!”. For example, in the Smart Ponzi which address is 0xbc3d0e2cf6720665c49455391b239160c89cdaa1 [16], it promise the security in line 37.

I.C.9. Social Network in solidity

Media_url is the url of the website, email, facebook and other social media url in the solidity code. We checked the solidity code and documented the url in the comment. solidity_url is a dummy variable that whether the contract have Media_url.

I.C.10. Key Words in Solidity code

Key_X is a dummy variable whether the solidity code contain the keyword. The key word include [”ponzi”, ”payout”, ”payment”, ”invest”, ”daily”, ”rate”, ”risk”, ”affiliate”, ”ref”, ”review”]

I.C.11. Key Words in Solidity code

Key_X is a dummy variable whether the solidity code contain the keyword. The key word include [”ponzi”, ”payout”, ”payment”, ”invest”, ”daily”, ”rate”, ”risk”, ”affiliate”, ”ref”, ”review”]

[16]https://etherscan.io/address/0x068abd01efff87943c6853abff3d20edfa9f9a18#code

I.C.12. Fee account

Fee account (Fee_account) is defined as the address that receive the commission fee including dev fee, adv fee, public fee and any other fee. The fee account included three sources: 1. The address already written in the solidity which cannot be changed. 2. The appointed address when contract is created and cannot be changed. 3. The address set by the owner after the contract deployed on the mainnet. In the case 3, the fee account can be changed. We checked the transaction history and found all fee account since the contract created.

I.C.13. Interval Between Withdrawals

Interval between withdrawals(withdraw_interval) refers to the minutes that investors have to wait since they can withdraw next time although there are enough money to distribute among them. For example, in the Smart Ponzi which address is 0x0689418b68122149a737a7cc7a4
[17], investors must wait 1440 minutes (1 day) to get interest until they invest it(line36-49).

I.C.14. Ref Ratio

Ref ratio(ref_ratio) refers to the percentage of investment of new investors that pay to their referrers. Besides the direct referrers which refer the investors, some Smart Ponzi may pay undirect referrers that refer the referrers of new investors. In this variable, we only calculated the ratio that only pay direct referrers. For example, in the Smart Ponzi which address is 0x0135c9a7bff72aa26e1d105ff5000e454e4dde7a [18], the Ref ratio is 7%(line 175-178)

I.C.15. Rebate Ratio

Rebate ratio(rebate_ratio) refers to the percentage of investment of new investors that pay back to them. For example, in the Smart Ponzi which address is 0x0135c9a7bff72aa26e1d105ff5000e4
[19], the Rebate ratio is 3.5%(line 180-183).

[17]https://etherscan.io/address/0x0689418b68122149a737a7cc7a49b2ad7c3049cc#code
[18]https://etherscan.io/address/0x0135c9a7bff72aa26e1d105ff5000e454e4dde7a#code
[19]https://etherscan.io/address/0x0135c9a7bff72aa26e1d105ff5000e454e4dde7a#code

I.C.16. Can Exit

Can exit(can_exit) is a dummy variable that Smart Ponzi give investors the choice to exit the scheme that they give up the opportunity of getting income anymore and get part of their investments. For example, in the Smart Ponzi which address is 0x3cad08c748fe69422e9e0b088a04 [20], investor can exit the smart contract by send transactions with specific value(0.0000012 ETher) (line 263-266), and smart contract will send part of his investment and delete his record(line 234-250).

I.C.17. Can Get Part Balance

Can get part balance(can_get_part_balance) is a dummy variable that investors can get part of their money when the balance of the Smart Ponzi is below the amount which is proposed to pay investors. For example, in the classic Ponzi which address is 0x3b3a608c676644959dde08fb252a7d64e71ac843 [21], if amount need to pay is not enough on the contract balance, balance will be sent what is left(line 33-35).

I.C.18. Can Reinvest

Can reinvest(can_reinvest) is a dummy variable that Smart Ponzi is able to reinvest the same scheme when they reach the profit limit or are deleted from the payment queue.

I.C.19. Fixed Revenue

Fixed Revenue(fixed_revenue) is a dummy variable that whether the payment from Smart Ponzi to the investors is fixed amount. For example, in the Smart Ponzi which address is 0x0444f06a52320af2df7e60d1923080002838ce93 [22], the revenue is fixed to the 1 Ether(line 63-66)

[20]https://etherscan.io/address/0x3cad08c748fe69422e9e0b088a04fe54a1e8fb7b#code
[21]https://etherscan.io/address/0x3b3a608c676644959dde08fb252a7d64e71ac843code
[22]https://etherscan.io/address/0x0444f06a52320af2df7e60d1923080002838ce93#code

I.C.20. the Limit of Withdrawal Time

The limit of withdrawal time(the_limit_of_withdrawal_time) is a dummy variable that is equal to 1 if investors is allowed to withdrawal for limited number of times such as once or twice. For example, in the Smart Ponzi which address is 0x173ee6e41bf96c0a1c58bc4c31699 916b10d7ef2 [23], investor are only allowed to withdraw for one time(line372-407).

I.C.21. Manaully Payment

Manaully payment(manaully_payment) is a dummy variable whether Smart Ponzi will not pay investors unless they send transactions to withdraw their interests. For example, in the Smart Ponzi which address is 0x02c60d28be3338014fef3fdf50a3218b946c0609 [24], investor has to manually send transaction to get his interests(line33-48)

I.C.22. Auto Payment

Auto payment(auto_payment) is a dummy variable whether the Smart Ponzi pay old investors automatically that they don't need to send transactions to withdraw their interests. For example, in the Smart Ponzi which address is 0x0475db744818f6f1a7224886a1b4927670790924 [25], investor may receive interests by other investors investment transaction(line148-178).

I.C.23. Balance Type

Balance type (ponzi_r_balance) is a dummy variable whether the interest rate of Ponzi is correlated to the contract' balance.

I.C.24. Investment Type

Investment type (ponzi_r_investment) is a dummy variable whether the interest rate of Ponzi is correlated to the investors' investment.

[23]https://etherscan.io/address/0x173ee6e41bf96c0a1c58bc4c31699916b10d7ef2#code
[24]https://etherscan.io/address/0x02c60d28be3338014fef3fdf50a3218b946c0609#code
[25]https://etherscan.io/address/ 0x0475db744818f6f1a7224886a1b4927670790924#code

I.C.25. Holdtime Type

Holdtime type (ponzi_r_holdtime) is a dummy variable whether the interest rate of Ponzi is correlated to the hold time that since the investors last withdraw of investment.

I.C.26. Participants Type

Participants type (ponzi_r_participants) is a dummy variable whether the interest rate of Ponzi is correlated to the contract' current investors.

I.C.27. The Lowest Interest

The lowest interest (lowest_interest) is the lowest daily interest rate of investors' investment.

I.C.28. Original (fun)

Original (fun) (original_fun) is a dummy variable whether the Smart Ponzi is original or copied from another Smart Ponzi classified by contract functions. First, we use the BOW(bag of words) model to encode the ABI of smart contract into the sparse vector. Then, we use Birch algorithm contract to calculate the similarity of different Smart Ponzis and cluster them. Finally, we got 196 clusters of smart Ponzis. In each cluster, we label the earliest created Smart Ponzis as original (Original_fun=1), else are labeled as copycats(Original_fun=0). Specially, if some Smart Ponzis are created in the same day which is the earliest day, we labeled them as original(Original_fun=1).

I.C.29. Original (name)

Original (name)(original_name) is a dummy variable whether the Smart Ponzi is original or copied from another Smart Ponzi classified by contract name. First, we download the contract name from the Etherscan.io. Then, we alter the contract name to the lower case and drop the stopwords in the contract name(stopwords include the number and punctuation). Third, we use the BOW(bag of words) model to encode the contract name into the sparse vector. Forth, we use Birch algorithm contract to calculate the similarity of different Smart Ponzis and cluster them. Finally, we got 215 clusters of smart Ponzis. In each group, we label the earliest created Smart Ponzis as original (Original_name =1), else are labeled as copycats(Original_name=0). Specially, if some Smart Ponzis are created in the same day which is the earliest day, we labeled them as original(Original_name=1).

I.D. Code Up Methods: Data crawler from other website

I.D.1. Bitcointalk posts

Have bitcointalk posts(have_bitcointalk) is a dummy variable whether there are posts in the bitcointalk.org mentioned the specific address of Ponzi Scheme.

I.E. Participants' Profit

I.E.1. Methods

These variables are calculated by the Smart Ponzi contracts' transaction history on Ethereum since they are created to the block 11000000(Oct 06, 2020). The failed transactions were removed to calculate these variables. User in this table refers to the participants, fee account or Ponzi contract creators.

I.E.2. User

Ethereum account address of user.

I.E.3. Address

Ethereum address of Ponzi contract.

I.E.4. Income

Income is the variable that calculated by sum the ETh income of user i of the specific contract j. It is defined as $Income_{i,j} = \sum_{0}^{T} Income_{i,j,t}$ Alternative variable is $Income\_usd_{i,j}$, which use the us dollar to calculate the variable.

I.E.5. Investment

Investment is the variable that calculated by sum the ETh investment of user i of the specific contract j. It is defined as $Investment_{i,j} = \sum_{0}^{T} Investment_{i,j,t}$ Alternative variable is $Investment\_usd_{i,j}$, which use the us dollar to calculate the variable.

I.E.6. Profit

Fee_profit is the variable that calculated by sum the ETh profit of user i of the specific contract j. It is defined as $Profit_{i,j} = \sum_{0}^{T} Income_{i,j,t} - \sum_{0}^{T} Investment_{i,j,t}$ Alternative variable is $Profit\_usd_{i,j}$, which use the us dollar to calculate the variable.

I.E.7. Rate_of_return

The variable Rate_of_return refers to the ascending block time rank of investors' rate of return. $Rate\_of\_return_{i,j} = \frac{Income_{i,j} - Investment_{i,j}}{Investment_{i,j}}$ Alternative variable is $Rate\_of\_return\_usd_{i,j}$, which use the us dollar to calculate the variable.

I.E.8. Rank_rate

The variable *rank_rate* refers to the descending rank of users' Rate_of_return. $quantile_{i,j}$ refers to the quantile of users' profit among all users of contract j. *rank_rate_usd* and *quantile_usd* use US dollar instead of ETH.

I.E.9. Count_TX

The variable Count_TX refers to the transaction time between user i of the specific contract j,include the income transaction and investment transaction.

I.E.10. First

The variable $first$ refers to the order of the first transaction time of user i amont all users in the contract j. For example, if $first_{i,j}$=3, means user i is the third earliest user to transaction with contract j.

I.E.11. Isfriend

Friend defined as the address that made transactions with the creator no later than the date the contract was created. $Isfriend\_c\_noponzi$ is the proxy of Isfriend in the main results. $Isfriend\_nosc$ is defined as whether user i is friend of creator or fee account of Smart Ponzi j, where user i is not a smart contract address. $Isfriend\_nop$ is defined as whether user i is friend of creator or fee account of Smart Ponzi j, where user i is not a Ponzi contract address. $Isfriend$ is defined as whether user i is friend of creator or fee account of Smart Ponzi j. $Isfriend\_c$ is defined as whether user i is friend of creator of Smart Ponzi j, where user i is not a Ponzi contract address. $Isfriend\_c\_neat$ is defined as whether user i is friend of creator of Smart Ponzi j, where user i is not a Ponzi contract address or creator or fee account of Smart Ponzi j.

I.E.12. Isfee

$Isfee$ is defined as whether user i is the account which collect fee of Smart Ponzi j.

I.E.13. Iscreator

$Iscreator$ is defined as whether user i is contract creator of Smart Ponzi j.

I.E.14. rank_block

The variable rank_block refers to the ascending block time rank of investors' first investment.