Detecting Ponzi schemes on Ethereum

Marion Le Tilly
Ecole polytechnique
Palaiseau, France
marion.le-tilly@polytechnique.edu

Ashish Gehani
SRI International
Menlo Park, California
gehani@csl.sri.com

Eunjin (EJ) Jung
University of San Francisco
San Francisco, California
ejung@cs.usfca.edu

ABSTRACT

Since 2009, the popularity of blockchain-based currencies, such as Bitcoin and Ethereum, has grown among enthusiasts. Relying on the anonymity provided by the blockchain, hustlers have adapted offline scams to this new ecosystem. As a result, Ponzi schemes are proliferating on Ethereum, dressed up as rewarding and secure investment schemes. They reward early investors with the investments of the next ones before collapsing, leaving the last investors empty handed. Illegal in the offline world, they are creating thousands of victims on Ethereum, while stealing millions of dollars worth of ether. Following several studies on Ponzi scheme behavior and detection, this project uses data mining to provide a detection model for Ponzi schemes on Ethereum. We built a dataset of Ethereum addresses, including known Ponzi scheme addresses, and designed features based on their compiled code and transactions. Using Weka to benchmark several classification algorithms, we obtained a model achieving high accuracy and high recall. It can be used as soon as a scheme is uploaded on the blockchain. Our approach provides perspectives on Ponzi scheme behavior and avenues to explore in future work.

CCS CONCEPTS

• **Computer systems organization → Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability;

ACM Reference format:
Marion Le Tilly, Ashish Gehani, and Eunjin (EJ) Jung. 2018. Detecting Ponzi schemes on Ethereum. In *Proceedings of ACM SAC Conference, Pau,France, April 9-13, 2018 (SAC'18),* 7 pages.
https://doi.org/xx.xxx/xxx_x

1 INTRODUCTION

In 2009, Satoshi Nakamoto came up with a protocol implementing global consensus among non trusted individuals: the Bitcoin [8]. A peer-to-peer network enabling financial exchanges between users, based on economic incentives rather than trusted authorities. It operates on the blockchain: a public ledger keeping track of all the transactions in an immutable way using ground-breaking algorithms such as the *proof-of-work*. While all transactions are recorded on the blockchain and retrievable by any Ethereum client, Bitcoin still ensures the anonymity of its users who identify themselves solely with public keys, signing transactions with associated private keys.

Benefiting from the privacy, criminals quickly took advantage of its popularity to rip off other users by creating multiple scams. Maria Vasek et al. present an empirical analysis of Bitcoin-based scams: in 2014, they represented at least $11 millions making 13,000 victims [10]. Unlike precedent innovations, cryptocurrencies are not yet subjects to any government regulations due to its lack of trusted authorities and its privacy, making it difficult to compensate the damages caused by frauds. For instance, the immutability of the blockchain prevents one self from ordering a chargeback for a transaction.

Most of the scams are inspired by offline precedents such as Ponzi schemes: a 150 year old deceit emulating an investment program with high returns, of which one can cash out only if there are enough new users to pay its profit with their investment. As a result, early investors cash off the backs of the most recent ones, who end up empty-handed. These scams damage the reputation of the whole cryptocurrencies ecosystem including Ethereum. Launched by Vitalik Butterin in 2015, its blockchain enables the use of smart contracts: pieces of code stored on the blockchain implementing applications capitalizing on distributed consensus as well as privacy, such as online voting, ownership of a physical device, currencies and financial assets. It focuses on the simplicity of its protocol to democratize cryptocurrencies potential, unfortunately it also attracts ill-intentioned users. Ponzi schemes are actually easy to implement on Ethereum which largely accounts for their growing number. In 2014, [10] shows that it already represents 60% of the frauds dollar volume and is expected to increase rapidly. In addition, successful Ponzi are proven to impact few users who make huge contribution, generating highly unfortunate victims. *Smart Ponzi* prey upon the transparency provided by the blockchain as a show of faith to draw inexperienced customers who can not figure out the flaws of both the deceit and its implementation.

In [3], Massimo Bartoletti analyzes Ponzi schemes behaviour on Ethereum, using similarities between contracts bytecode to locate 191 of them. In total, the contracts collected almost half a million USD from more than 2000 distinct users. [3] highlights characteristics of Ponzi schemes behaviour such as a high gini coefficient, a measure of inequality in the distribution of money made by contracts to investors. [4] reuses this characteristics to apply data mining techniques to Ponzi schemes implemented on Bitcoin. Their best classifier manages to detect 31 Ponzi schemes out of 32 with 1% of false positives. It detects Ponzi schemes using features computed over a contract's lifetime, narrowing the range of countermeasures that can be implemented against such schemes because tokens can not be refunded nor flagged as stolen.

Nevertheless, its approach has inspired [6] which describes a similar approach but includes features based on the contract's code

SAC'18, April 9-13, 2018, Pau,France

ACM ISBN 978-1-4503-5191-1/18/04…$15.00
https://doi.org/xx.xxx/xxx_x

which is stored on the blockchain. Their model classifies correctly 81% of the smart Ponzi.

Similarly, this project provides a model detecting smart Ponzi as soon as their code is uploaded on the blockchain by the Ethereum client, in order to prevent them from scaling or from gaining popularity by flagging them as malicious investment schemes. It classifies correctly 98% of smart Ponzi. To this end, it proposes an approach to build a dataset and to select features with regards to their robustness and their relevance in a rapidly evolving context such as the cryptocurrencies. Using Weka[1], a data mining software, it investigates several classifying algorithms with regards to metrics cogent with fraud detection. Finally, it suggests a model consistent with expectations and hypotheses raised by the elaboration of the latter perspectives and brings forward avenues to improve the overall relevance of the project. It takes place as an initial approach within the context of early fraud detection on cryptocurrencies.

2 BACKGROUND

This section elaborates on Ethereum smart contracts and how Ponzi schemes exploit their characteristics.

A blockchain system is essentially a state transition and maintenance system. For Ethereum, a state defines an account and is stored on the blockchain, while transitions represent direct transfer of value or information (called payload) between states [7]. Each account has a unique 20-bytes address pointing at four fields used to guarantee the uniqueness of each of its state:

- the **nonce**: incremented at each transaction to ensure it is processed only once;
- the account **balance**: current amount of ether of the account;
- the **contract bytecode**: immutable because it is stored on the blockchain;
- the account **storage**.

An account is said to be externally owned if it is controlled by a private key as opposed to contract accounts controlled by their contract code. Each time a contract account receives a transaction, it activates itself using the payload as input: reading and writing to internal storage, sending transactions or creating contracts in turn. For instance, when a smart Ponzi receives a transaction containing a sufficient amount of ether, it triggers payment transactions to previous participants. In this context, the transaction received by the smart Ponzi is called a normal transaction as opposed to the one it fires called internal transactions.

To broadcast transactions to the whole network, they are packaged into blocks which are executed and verified by miners. They are encouraged to do so with a fee collected for each executed transaction. Fees are proportional to the computation required for the execution of the transaction, they are calculated in *gas*[2]. The account firing the transaction defines the *gas-price* which, multiplied by the number of gas, constitutes the reward of the miners. The higher the price, the more eager miners will be to execute the transaction. If there is not enough gas, the execution fails, it *runs out of gas*: all side effects are reversed but the fee is lost. After the execution, transactions are validated block by block using a consensus protocol similar to Bitcoin, ensuring the safety of the system unless an attacker manages to own more than 51% of the computing power of the whole network.

To enable the implementation of blockchain applications, Ethereum integrates an Ethereum Virtual Machine (EVM), a run-time environment for EVM bytecode, to its platform. Thus, creating an Ethereum application boils down to writing its source code in a high-level language such as Solidity[3], compiling it into EVM bytecode and eventually uploading it on the Ethereum blockchain. Once the contract has been uploaded, it is active: it can fired and receive transactions but most importantly, its bytecode is immutable.

Smart Ponzi capitalize on this immutability to attract users, claiming that because the contract will run forever without being interfered with, investors are assured to make a profit. Yet, they fail to mention that creators have the right to modify its parameters such as the fee required for each investment or that, as explained in [3], code are not proven to be without flaws. Briefly, immutability of the blockchain only ensure that the code execution is automatically enforced.

3 CLASSIFICATION MODEL

3.1 Dataset aggregation

First of all, a dataset of Ethereum smart contracts is required to feed the classifier. For fraud detection studies, it often is the most laborious step, requiring strenuous manual verification and research to distinguish fraud and non-fraud instances. Thankfully [3] provides an open-source dataset containing 150 Ponzi scheme addresses, available at [2]. Each address has been manually inspected to confirm whether it is a Ponzi scheme by both [3, 6], thus they are considered as ground truth. A scrapper provided an extra 3314 addresses of smart contracts verified by Etherscan [1]. In a few words, when a source code is verified it proves the code has deterministic and verifiable builds, for users it appears as a vote of confidence in the code. However, it does not prevent the code from having flaws. The 3314 contract addresses were considered as non Ponzi by default, yet according to [6] at least 0.15% of all the smarts contracts of Ethereum are Ponzi schemes, meaning that at least one of them is a Ponzi scheme. This trait will have its relevance during the classification process. The intersection of these two sets of addresses is empty since the non Ponzi addresses have been generated from September 2017 to June 2018 while the Ponzi set has been generated before August 2017.

For each address in the dataset, the Etherscan APIs have been used to extracts information concerning transactions received from or emitted by it. Then through Web3 python interface [4], the EVM bytecode was recovered from the blockchain.

3.2 Classification models

In order to come up with a classification model as efficient as possible, several well-known classification algorithms competed representing different learning strategies:

- **J48**: a decision tree which enables visualization of the decision process exhibiting which features are used and how;

[1] https://www.cs.waikato.ac.nz/ml/weka/
[2] a measurement unit in Ethereum
[3] http://solidity.readthedocs.io/en/develop/
[4] github repository: https://github.com/ethereum/web3.py

- **Random Forest** (RF): a decision forest, it aggregates the prediction of individual trees. It has proven to be an excellent classifier for previous fraud detection issue [5];
- **Stochastic Gradient Descent** (SGD): iterative method for optimizing a differentiable objective function.

Since we aim to detect Ponzi schemes, models will be evaluated according to two main evaluation metrics:

- **precision**: the ratio of instances correctly classified.

$$precision = \frac{TP}{TP + FP}$$

- **recall**: the ratio of Ponzi instances correctly classified.

$$recall = \frac{TP}{TP + FN}$$

Actually, since the model is supposed to detect Ponzi schemes, false positives are a second-order issue: they might even be true positives considering the non Ponzi class is composed of random contracts. On the contrary, false negatives need to be penalized, it means the model fails to detect smart Ponzi, hence the emphasis on the recall.

What is more, the analysis has to take into account that the dataset is excessively unbalanced with 3314 non Ponzi instances and only 145 Ponzi instances. As a result, the *ZeroR* algorithm classifies correctly 95.8% of the instances by classifying every instances as non Ponzi. This is the reason why, each algorithm is trained on two versions of the dataset:

- raw dataset;
- undersampled dataset: a dataset for which the non Ponzi class is subsampled to be twice the size of the Ponzi class;

Other filters such as modifying the weights of both classes or oversambling the Ponzi class have been experimented without success.

Each model will be trained using a cross-validation process with 10 folds. Metrics may vary with the random seed used for the cross-validation, therefore performances will be computed as an average.

3.3 Features design and robustness

Features design is a crucial upstream work to have an efficient classification model. The following approach is motivated by previous works [3, 6, 9, 11] which shed light on two types of features. Both [3, 6] granted a significant share of their work to the contracts bytecode to create features or to find similar contracts, while [4, 6, 9, 11] characterized contracts by their interactions with their environment.

3.3.1 Transactions features. Computing features using the interactions of the smart Ponzi with its users is a logical and relevant idea considering that is how real life Ponzis are identified.

Basic features. The literature [4, 6, 9, 11] provides a range of basic features which characterize smart contracts behavior:

- **nbr_tx_in** (resp. **nbr_tx_out**): the number of transactions which transferred ether to (resp. from) the smart contract;
- **Tot_in** (resp. **Tot_out**): the total amount of ether transferred to (resp. from) the smart contract;
- **mean_in** (resp. **mean_out**): the average value of ether transferred to (resp. from) the smart contract;
- **sdev_in** (resp. **sdev_out**): the average value of ether transferred to (resp. from) the smart contract;

Robustness: A contract creator can easily tamper with such features using well-known methods such as coinjoin[5] to run interference. It has become mainstream to use several public addresses to receive and proceed to payments. Only the total amount of money going in and out of the contract can not be corrupted.

Extended features Inspired by their behavior, the following features focus on smart Ponzi specificity in comparison with the average smart contract:

- **avg_time**: average amount of time between two transactions;
- **lifetime**: time between the first and the last transactions;
- **gini_in** (resp. **gini_out**): the gini coefficient computed over the value of transactions to (resp. from) the smart contract;
- **gini_time**: the gini coefficient computed over the time of the transactions from the smart contracts.

Ponzi schemes are considered as fraud mainly because the sooner a user invests, the bigger the reward, to the extent that some investors never see their money back. Early investors receive huge amount of money while last investors receive very limited ones. This distribution of money is unequal and therefore characterized by a high gini coefficient. The gini coefficient, ranging from 0 to 1, characterizes inequality among a given distribution: the closer it is to 1 the more unequal the distribution is. For a given distribution a among n participant, it is computed through the following formula:

$$G(a) = \frac{\sum_{i=1}^{n} (2i - n - 1)a(i)}{n \sum_{i=1}^{n} a(i)}$$

Furthermore, smart Ponzis have a short lifetime, but make many transactions [3], explaining the use of the *avg_time* feature. Even better, as they grow old, the frequency at which they issue their transactions slows, hence the *gini_time* feature.

Robustness: as stated above, these transactions are as robust as the ability to address methods used to run interference about interactions between users. In our case, we made the hypothesis that the use of such techniques on Ethereun is unusual enough to consider these features as robust.

3.3.2 Code features. Using evmdis[6], an open source disassembler, each contract bytecode has been turned into its equivalent in opcodes. Opcode is the abbreviation of operation code, it is a portion of a machine language that tells the CPU what operations needs to be done. Figure 1 is a portion of disassembled bytecode from a smart Ponzi.

Then, the frequency of each opcode in the contract code is computed and stored in the database. The size of the bytecode of each contract in bytes is stored too to create the **size_info** feature. This kind of features dissociate smart Ponzi from real life Ponzi since they do not have source code or open source automatic execution. What is more, using bytecode to compute features enables the detection of a Ponzi scheme before it operates.

Robustness: By characterizing the implementation of the Ponzi schemes, these features are difficult to manipulate for hustlers to

[5] https://en.bitcoin.it/wiki/CoinJoin
[6] github repository: https://github.com/Arachnid/evmdis

```
# Stack: []
0x4     MSTORE(0x40, 0x60)
0xA     JUMPI(:label0, !CALLDATASIZE())

# Stack: []
0x13    PUSH(CALLDATALOAD(0x0) / 0x2 ** 0xE0)
0x19    DUP1
0x1E    JUMPI(:label1, POP(@0x13) == 0x72EA4B8C)

:label0
# Stack: []
0x20    PUSH(:label2)
0x23    PUSH(CALLVALUE())
0x24    PUSH(0x0)
0x26    DUP1
0x27    DUP1
0x31    DUP4
0x36    JUMPI(:label6, POP(@0x23) < 0xDE0B6B3A7640000)

# Stack: [0x0 0x0 0x0 @0x23 :label2]
0x37    PUSH(0x1)
0x39    DUP1
0x3B    DUP1
0x3C    PUSH(POP(0x1) + SLOAD(POP(0x1)))
0x3D    SWAP1
0x3E    DUP2
0x3F    SWAP1
0x40    SSTORE(POP(0x1), POP(@0x3C))
0x41    PUSH(0x3)
0x43    DUP1
0x44    PUSH(SLOAD(POP(0x3)))
0x45    PUSH(CALLER())
0x46    SWAP2
0x47    SWAP1
0x48    DUP2
0x4E    JUMPI(0x2, !(POP(@0x3C) < POP(@0x44)))
```

Figure 1: Disassembled bytecode for contract 0x109c4f2ccc82c4d77bde15f306707320294aea3f

elude classification. Besides, dynamic analysis could reinforce their robustness by providing only bytecode relevant to the contract execution, dumping bytecode used to *jam* the classifier.

4 MODELS PERFORMANCE

This section presents our two-steps-approach and an in-depths analysis of the features used in the experiments.

4.1 Initial model

We initially used basic features inspired by how smart contracts operates on the blockchain, combined with code features to classify smart contracts. The use of this particular set of features is closer to a classification process rather than a detection one.

Table 1: Initial model performance comparison

Model	Filters	Accuracy	Ponzi Recall
J48	raw	0.996	0.959
	Subsampling	0.977	0.966
Random Forest	raw	0.993	0.841
	Subsampling	0.982	0.959
SGD	raw	0.993	0.897
	Subsampling	0.935	0.955

Table 2: Initial model confusion matrices, without subSampling

CV-10	Predicted	
RF	**P**	**nP**
P	122	23
nP	0	3314

CV-10	Predicted	
J48	**P**	**nP**
P	139	6
nP	6	3308

In Table2, the Random Forest model classifies 15% of the Ponzi schemes as non Ponzi and J48 only 4%. While RF has a better accuracy, it also has a penalizingly low recall, which makes J48 the more relevant classifier for this approach, with a recall 0.966 when a SubSampling filter is applied.

4.2 Complete model

The second approach uses both the basic and the extended set of features combined with code features.

Table 3: Complete model performance comparison

Model	Filters	Accuracy	Ponzi Recall
J48	raw	0.997	0.966
	Subsampling	0.981	0.979
Random Forest	raw	0.993	0.841
	Subsampling	0.982	0.966
SGD	raw	0.992	0.883
	Subsampling	0.966	0.945

Table 4: Confusion Matrices, with SubSampling

CV-10	Predicted	
RF	**P**	**nP**
P	140	5
nP	0	290

CV-10	Predicted	
J48	**P**	**nP**
P	143	2
nP	2	288

In comparison with the first step, Table **??** shows how accuracy and recall increased for all models. Besides, the same trend can be observed: J48 has the best trade-off with a recall of 0.972 using a SubSampled dataset. The visualization of the decision tree in 2 illustrates how it uses both code and transactions features. Yet, with a different cross-validation seed, Figure 3 shows a decision tree using only code features.

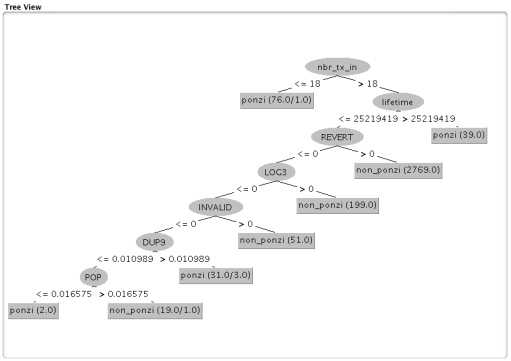


Figure 2: Complete model: J48 Decision Tree (seed of 1

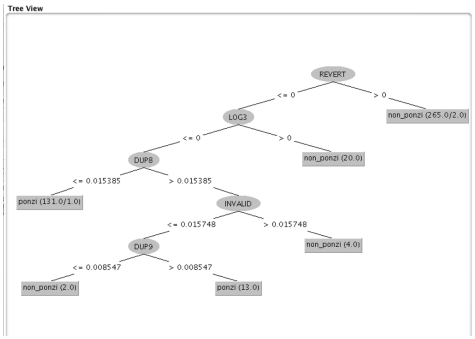


Figure 3: Complete model: J48 Decision Tree (seed of 4)

4.3 Complete model feature strength

In order to evaluate our results, a critical appreciation of the features is required. A feature is relevant to the model if it is useful, meaning it actually provides information to the classifier.

To measure features strength, we ranked our features selection using Weka attributes selection: features are ranked according to their information gain (IG)[7] and chi-square (CS)[8].

Feature	Information gain	Chi-square
lifetime	0.146	1769
nbr_tx_in	0.121	1599
CODECOPY	0.120	1323
REVERT	0.139	985
size_info	0.106	984
avg_time	0.0633	986
gini_time	0.0467	492

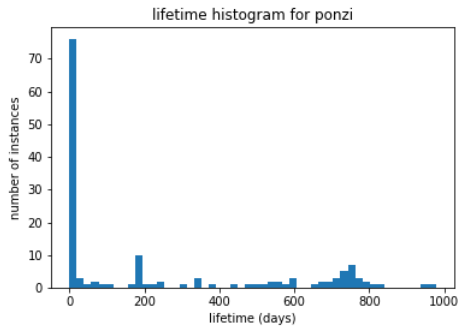
Table 5: Feature Comparison

In Table 5, both metrics give almost the same ranking as top features: two code and six transaction features. Table 6 gives a lead to explain why these features are discriminatory. In the following sections, we discuss each feature's usefulness.

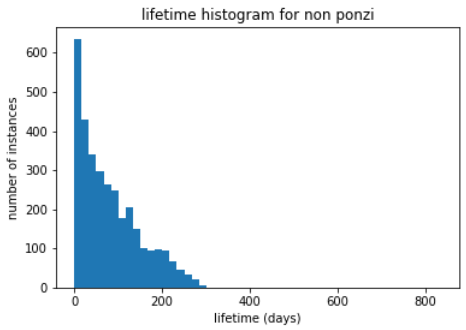
Feature	Ponzi	non Ponzi
lifetime	232 days	82 days
nbr_tx_in	108	1509
CODECOPY	0.041	0.002
REVERT	0.00064	0.041
avg_time	125 hour	15 hours
gini_time	0.0003354	0.0001022

Table 6: Features average

4.3.1 lifetime. In both cases, lifetime seems to be the most significant feature. Table 6 shows a serious gap between the average lifetime of Ponzi schemes and random contracts. It can be explain by the way the dataset was collected: the oldest non Ponzi contract is less than a year old while some of the smart Ponzi are two to three years old. Figure 4 shows lifetime histograms for both instances. While random contracts have a gaussian distribution with a median of 64 days, Ponzi scheme have a median of 9 days.



(a) Smart Ponzis



(b) Non Ponzis

Figure 4: Lifetime histograms

However, in practice this feature is not as helpful as it seems since, it requires the contract to have ceased its activity before being computed. Thus, it is useless for an early detection approach.

4.3.2 nbr_tx_in. As for the lifetime, when aggregating the dataset, the non Ponzi instances were collected with a threshold of 10 transactions minimum, to ensure the contract was active. Yet,the most concerning issue is the strength of this feature which is the first node of the decision tree, as shown in Figure 2. Actually, with the rise of cryptocurrencies, Ponzi schemes are expected to increase in terms of audience for instance. Its strength in our model could prevent it from scaling to bigger Ponzi schemes.

4.3.3 CODECOPY. This instruction is used to copy the runtime part of the contract in EVM's memory, while it has a median of 0 for non Ponzi, Figure5 shows it has a significant frequency in some of the smart Ponzis.

4.3.4 REVERT. On the contrary, while smart Ponzi have a median of 0, an important subset of non Ponzi contracts uses this instruction. It is related to the modularity of the code, and used by people who want to limit gas money, which seems not to be the case of Ponzi schemes creators.

[7] https://en.wikipedia.org/wiki/Information_gain_ratio
[8] http://weka.sourceforge.net/doc.stable/weka/attributeSelection/ChiSquaredAttributeEval.html

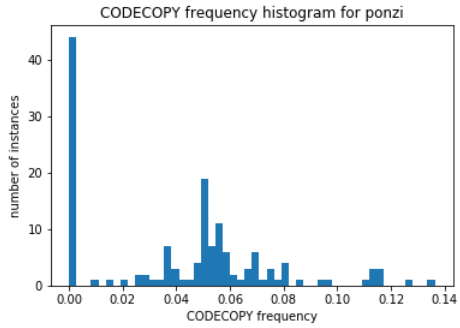


Figure 5: CODECOPY frequency histogram for smart Ponzis

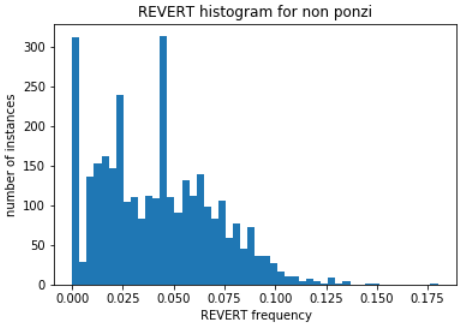
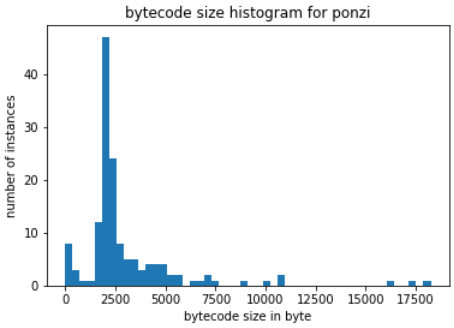
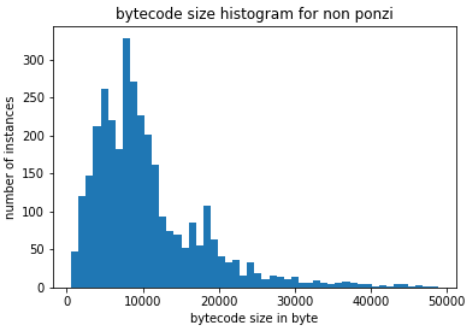


Figure 6: REVERT frequency histogram for non Ponzi



(a) Smart Ponzis



(b) Non Ponzis

Figure 7: Bytecode size histograms

4.3.5 size_info. Figure 7 illustrates the diversity of non Ponzi contracts compared to the similarity of the smarts Ponzi with a standard deviation 2.5 times smaller than the non Ponzi.

4.3.6 avg_time. Table 6 shows a deep contradiction with the intense activity of the Ponzi schemes described in [3] with an average ten times bigger for smart Ponzi. Yet, Ponzi schemes have a median of 5.3 hours, compared to 4.3 hours for non Ponzis. It is explains by some Ponzi schemes firing transactions a year after their last sign of activity, it would not be taken into account with an early detection approach. Besides, the first quartile of smart Ponzi is smaller than 6 minutes compared to 70 minutes for non Ponzi, an observation that supports [3]. An early detection approach combined with stream mining would make the best out of this feature.

4.4 Early detection

Given the strength of some of the transactions features, it is logical to assume the performances of the model partially depends on them, or even that they could increase over a contract lifetime. That is the reason why, the complete model with the J48 algorithm has been trained on several subsampled datasets of which the features had been computed at different times: when the contract is uploaded on the blockchain, after two days of activity, four days etc. As for the previous performances, metrics have been computed as the average of the metrics obtained with several cross-validation seeds.

The Figure 8 illustrates the performances evolution over a contract lifetime. We could have expected a slight increase after the first

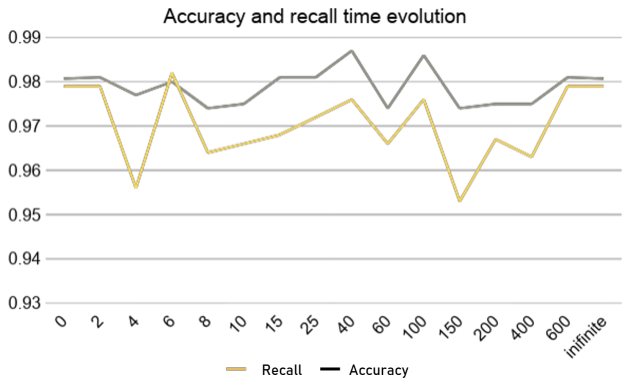


Figure 8: Time evolution of J48 performances

days of activity, thanks to the growing strength of the transaction features. Yet, the variations have a very little standard deviation with minimums above 0.95 for the recall and above 0.97 for the accuracy.

Eventually, the best results are given by a model trained only using code features, computed during day zero, when the contract is uploaded on the blockchain. It enables early detection of the smart Ponzi which makes possible the implementation of a flagging system. As soon as a suspicious contract is uploaded on the

blockchain it could be flagged by our model to raise awareness among its potential users about the financial risks of investing in it.

5 FUTURE WORK

5.1 Address clustering

Thanks to our study of the time evolution of J48 performances, it appears that transaction features have less impact than code features. Their relevance could be improved if they could take into account the representation of a user. In other words, our current model assumes that each public key is a different user, while real users owns several public keys in their wallet and use them to protect their identity. Being able to identify a user's wallet of public keys would enable the computation of precise transaction features such as the lost or the profit of investors besides improving the latter ones. Aside from enhancing the performances of our current model, it would strengthen the overall robustness of the set of features: these features don't depends on the contract but on how users interact with it.

5.2 In practice...

Using our model under actual conditions, to detect unknown Ponzi schemes on the blockchain, is the next logical step of this project. It requires to compute features for the thousands of contracts on the blockchain which is computationally speaking demanding. Yet, using only code features would lighten the computational load and would provide some preliminary results. Even though the ratio of false positive is low for the J48 model with subsampling, with thousands of addresses, manual verification would still be strenuous.

6 CONCLUSION

Even thought the blockchain is considered as the next digital revolution, it is too often perceived as an unsecured payment mechanism due to its exploitation by hustlers. Victim of misinformation, the general public hears of the blockchain through scandal such as the DAO attack in 2016 instead of its potential to decentralize both services and currencies. Similarly, the proliferation of smart Ponzi, a hazardous investment scheme on Ethereum, contributes to undermine the public trust by making thousands of victims. Smart Ponzi are presented as brilliant and safe investment opportunities, while the risks involved are barely mentioned. That is why, we build a data mining model to enforce automatic detection of smart Ponzi as soon as they are uploaded on the blockchain. Indeed, our best classifier has a recall of 0.98, and uses features build on disassembled bytecode extracted from the Ethereum blockchain.

Eventually, this model could be used to flag suspicious contracts to provide information to potential investors about the risks they take by investing in a smart Ponzi. By testing it on the thousands of contracts already on the blockchain, it could provide a lower bond of the number of smart Ponzi as well as their impact on Ethereum.

ACKNOWLEDGMENTS

The authors would like to thanks Massimo Bartoletti for sharing [6].

The authors would like to thank Howon Song for providing an Etherscan scrapper which output the 3314 smart contracts addresses used in the dataset.

REFERENCES

[1] [n. d.]. The Ethereum block explorer. https://etherscan.io/. ([n. d.]).
[2] Massimo Bartoletti, Salvatore Carta, Tiziana Cimoli, and Roberto Saia. 2017. Dataset of Ponzi Schemes. goo.gl/CvdxBp. (2017).
[3] Massimo Bartoletti, Salvatore Carta, Tiziana Cimoli, and Roberto Saia. 2017. Dissecting Ponzi schemes on Ethereum: identification, analysis, and impact. *CoRR* abs/1703.03779 (2017). arXiv:1703.03779 http://arxiv.org/abs/1703.03779
[4] Massimo Bartoletti, Barbara Pes, and Sergio Serusi. 2018. Data mining for detecting Bitcoin Ponzi schemes. *CoRR* abs/1803.00646 (2018). arXiv:1803.00646 http://arxiv.org/abs/1803.00646
[5] Siddhartha Bhattacharyya, Sanjeev Jha, Kurian K. Tharakunnel, and J. Christopher Westland. 2011. Data mining for credit card fraud: A comparative study. *Decision Support Systems* 50 (2011), 602–613.
[6] Weili Chen, 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 (WWW '18)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 1409–1418. https://doi.org/10.1145/3178876.3186046
[7] Vitalik Buterin et al. 2014. A next generation smart contract and decentralized application platform. *Ethereum white paper* (2014).
[8] Satoshi Nakamoto. 2009. Bitcoin: A peer-to-peer electronic cash system,âĂİ http://bitcoin.org/bitcoin.pdf. (2009).
[9] Thai Pham and Steven Lee. 2016. Anomaly Detection in Bitcoin Network Using Unsupervised Learning Methods. *CoRR* abs/1611.03941 (2016). arXiv:1611.03941 http://arxiv.org/abs/1611.03941
[10] Marie Vasek and Tyler Moore. 2015. There's No Free Lunch, Even Using Bitcoin: Tracking the Popularity and Profits of Virtual Currency Scams. In *Financial Cryptography and Data Security*, Rainer Böhme and Tatsuaki Okamoto (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 44–61.
[11] Deepak Zambre. 2013. Analysis of Bitcoin Network Dataset for Fraud.