IMT SCHOOL FOR ADVANCED STUDIES LUCCA

Sharpening Ponzi Schemes Detection on Ethereum with Machine Learning

Letterio Galletta and Fabio Pinelli

DLT 2023

Bologna, May 25th 2023

Ponzi Scheme detection on Ethereum

What is a Ponzi scheme?

- It is an investment fraud
- It requires a constant flow of money from new investors to continue
- It inevitably collapses, when it becomes difficult to recruit new investors

There are smart contracts on Ethereum that are Ponzi Schemes.

Our work

Our main contributions:

- A **reusable and publicly available data set** that collects 4422 real-world smart contracts (3749 not Ponzi, and 673 Ponzi);
- A **binary classifier** to detect smart Ponzi contracts that performs better than classifiers proposed in the literature;
- The study of the impact of such features on the classification using **eXplainable AI techniques**;
- A small **set of features** that ensures a good classification quality.

https://github.com/fpinell/ponzi_ml

IMT SCHOOL FOR ADVANCED STUDIES LUCCA

The dataset

- Datasets with 4422 real-world smart contracts (3749 not Ponzi, and 673 Ponzi)
- Enriched the set of features starting from different sets available in the literature
- Collect the data for all the smart contracts using https://etherscan.io

	D1	D2	D3
Address	✓	✓	✓
Balance	✓	✓	✓
Lifetime	✓	✗	✓
Tx_in	✓	✗	✓
Tx_out	✓	✗	✓
Investment_in	✓	✓	✓
Payment_out	✓	✓	✗
#addresses_paying_contract	✓	✗	✓
#addresses_paid_by_contract	✓	✗	✓
Mean_v1	✓	✓	✓
Mean_v2	✓	✓	✓
Sdev_v1	✓	✓	✓
Sdev_v2	✓	✓	✓
Paid_rate	✓	✓	✓
Paid_one	✓	✓	✓
Known_rate	✓	✓	✓
N_maxpayment	✓	✓	✓
Skew_v1	✓	✓	✓
Skew_v2	✓	✓	✓
Investment_in/tx_in	✓	✗	✓
Payment_out/tx_out	✓	✗	✓
Percentage_some_tx_in	✓	✗	✓
Sdev_tx_in	✓	✗	✓
Percentage_some_tx_out	✓	✗	✓
Sdev_tx_out	✓	✗	✓
Initiator_gets_eth_Wo_investing	✓	✗	✓
Initiator_gets_eth_investing	✓	✗	✗
Initiator_no_eth	✓	✗	✗

Research questions

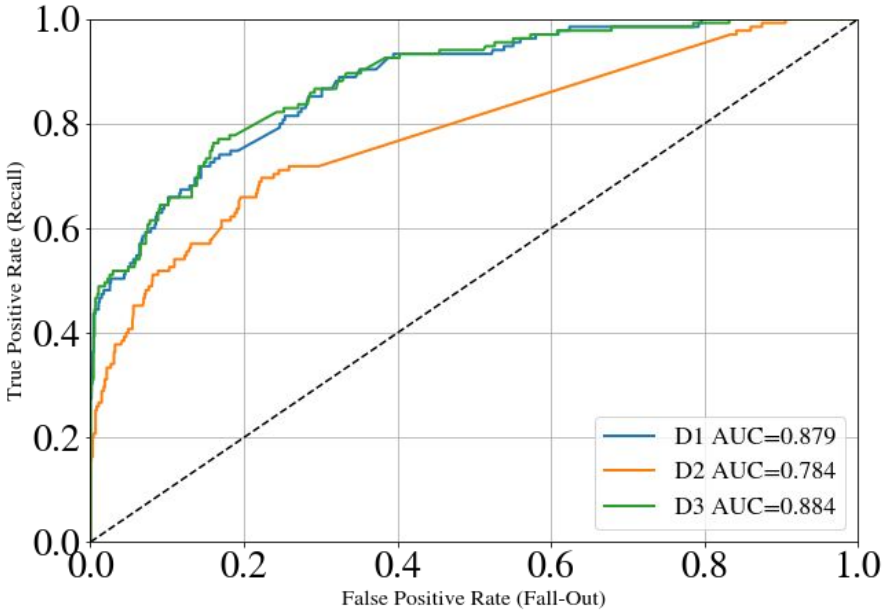
RQ1: Does the new features improve classification?

RQ2: What is the smallest and best set of features?

RQ3: What are the most important features and what is their contribution?

IMT SCHOOL FOR ADVANCED STUDIES LUCCA

RQ1: Does the new features improve classification?



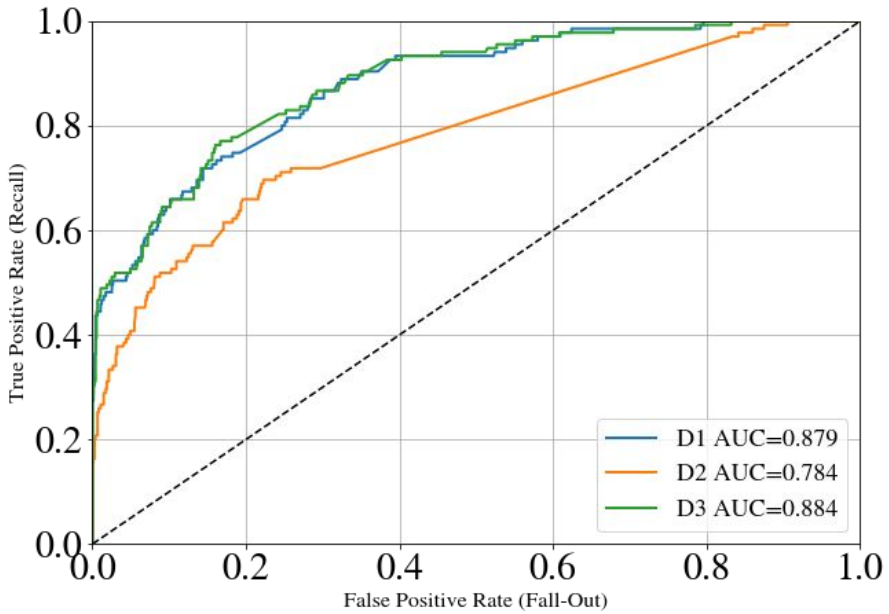
Data set	Metric / Classifier	Accuracy	AUC	F1	Precision	Recall
D1	Decision Tree	0.855	0.733	0.496	0.529	0.467
	LGBM	**0.904**	**0.883**	**0.608**	**0.805**	**0.489**
	Random Forest	0.896	0.875	0.562	0.787	0.437
D2	Decision Tree	0.861	0.674	0.481	0.559	0.422
	LGBM	**0.876**	**0.788**	**0.444**	**0.698**	**0.326**
	Random Forest	0.873	0.770	0.462	0.658	0.356

The best classifier is LGBM and it performs better on dataset D1 than D2

IMT SCHOOL FOR ADVANCED STUDIES LUCCA

RQ2: What is the smallest and best set of features?

	D1	D2	D3
Address	✓	✓	✓
Balance	✓	✓	✓
Lifetime	✓	✗	✓
Tx_in	✓	✗	✓
Tx_out	✓	✗	✓
Investment_in	✓	✓	✓
Payment_out	✓	✓	✗
#addresses_paying_contract	✓	✗	✓
#addresses_paid_by_contract	✓	✗	✓
Mean_v1	✓	✓	✓
Mean_v2	✓	✓	✓
Sdev_v1	✓	✓	✓
Sdev_v2	✓	✓	✓
Paid_rate	✓	✓	✓
Paid_one	✓	✓	✓
Known_rate	✓	✓	✓
N_maxpayment	✓	✓	✓
Skew_v1	✓	✓	✓
Skew_v2	✓	✓	✓
Investment_in/tx_in	✓	✗	✓
Payment_out/tx_out	✓	✗	✓
Percentage_some_tx_in	✓	✗	✓
Sdev_tx_in	✓	✗	✓
Percentage_some_tx_out	✓	✗	✓
Sdev_tx_out	✓	✗	✓
Initiator_gets_eth_Wo_investing	✓	✗	✓
Initiator_gets_eth_investing	✓	✗	✗
Initiator_no_eth	✓	✗	✗



D3 is quite similar to D1: we remove three features that may mislead the classifier

IMT SCHOOL FOR ADVANCED STUDIES LUCCA

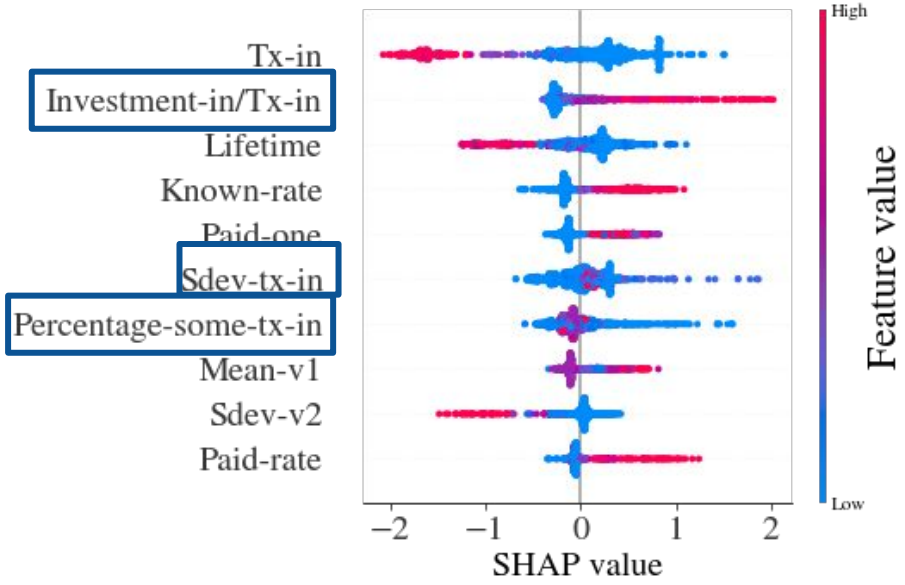
RQ3: What are the most important features?

- SHAP values (SHapley Additive exPlanations) is a method based on **cooperative game theory**

- SHAP shows the **contribution** or the importance of each feature on the prediction of the model

- The **beeswarm plot** is designed to display an information-dense summary of how the top features in a dataset impact the model's output

- 3 new features are in the set of the 10 most important features



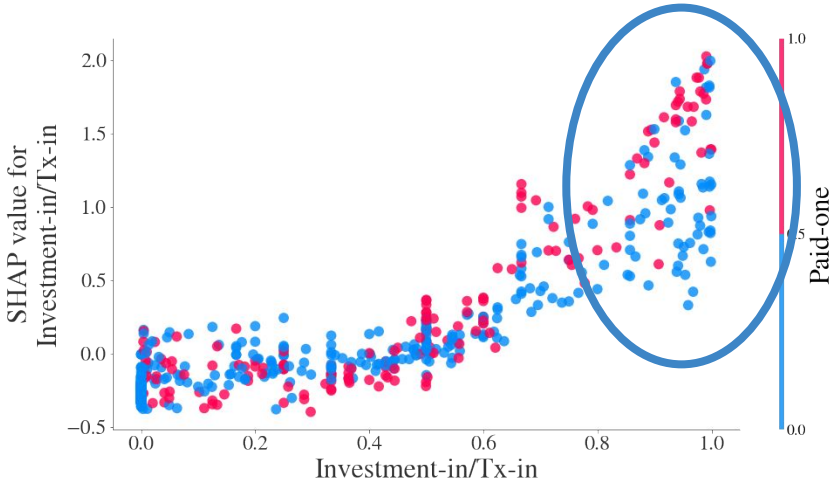
IMT SCHOOL FOR ADVANCED STUDIES LUCCA

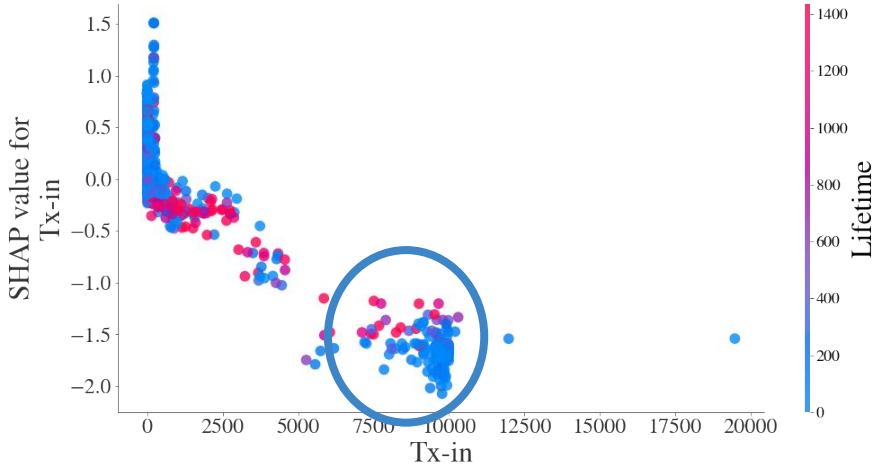
RQ3: What are the most important features?

Dependence plots between pair of features:

- Investment-in/Tx-in and Paid-one
- Tx-in and Lifetime

- The shap values (y-axis) varying as the colours of the second feature





Conclusions

- A **reusable and publicly available data set**
- A **binary classifier** to detect smart Ponzi contracts
- A small **set of features** that ensures a good classification quality
- The study of the impact of features using **eXplainable AI techniques**

Ongoing/future research activities:

- Identify features of bytecode
- Can we use only bytecode features?
- Study the robustness of the classifier

IMT SCHOOL FOR ADVANCED STUDIES LUCCA

Sharpening Ponzi Schemes Detection on Ethereum with Machine Learning

Letterio Galletta

letterio.galletta@imtlucca.it

Smart Ponzi

```
contract Multiplier {
    //Address of the "promoter" of the contract: she receives fee for each transaction
    address constant private PROMO = 0x5D5fe29339592eEb51c43E54F0a81cA7642B6d2b;
    //Percent received by the "promoter"
    uint constant public PROMO_PERCENT = 7;
    //How many percent for your deposit to be multiplied
    uint constant public MULTIPLIER = 121;

    //The deposit structure holds all the info about the made deposits
    struct Deposit {
        address depositor; //The depositor address
        uint128 deposit;   //The deposit amount
        uint128 expect;    //How much we should pay out (initially it is 121% of deposit)
    }

    Deposit[] private queue;  //The queue of investors
    uint public currentHead = 0; //The index of the first depositor in the queue.

    //This function receives all the deposits stores them  in the queue and make immediate payouts
    function () public payable {
        if(msg.value > 0){
            //Add the investor into the queue. Mark that he expects to receive 121% of deposit back
            queue.push(Deposit(msg.sender, uint128(msg.value), uint128(msg.value*MULTIPLIER/100)));

            //Send some promo to enable this contract to leave long-long time
            uint promo = msg.value*PROMO_PERCENT/100;
            PROMO.send(promo);

            //Pay to first investors in line
            pay();
        }
    }
```

Smart Ponzi

```
contract Multiplier {
    //Address of the "promoter" of the contract: she receives fee for each transaction
    address constant private PROMO = 0x5D5fe29339592eEb51c43E54F0a81cA7642B6d2b;
    //Percent received by the "promoter"
    uint constant public PROMO_PERCENT = 7;
    //How many percent for your deposit to be multiplied
    uint constant public MULTIPLIER = 121;

    //The deposit structure holds all the info about the made deposits
    struct Deposit {
        address depositor; //The depositor address
        uint128 deposit;   //The deposit amount
        uint128 expect;    //How much we should pay out (initially it is 121% of deposit)
    }

    Deposit[] private queue;  //The queue of investors
    uint public currentHead = 0; //The index of the first depositor in the queue.

    //This function receives all the deposits stores them  in the queue and make immediate payouts
    function () public payable {
        if(msg.value > 0){
            //Add the investor into the queue. Mark that he expects to receive 121% of deposit back
            queue.push(Deposit(msg.sender, uint128(msg.value), uint128(msg.value*MULTIPLIER/100)));

            //Send some promo to enable this contract to leave long-long time
            uint promo = msg.value*PROMO_PERCENT/100;
            PROMO.send(promo);

            //Pay to first investors in line
            pay();
        }
    }
```

Smart Ponzi

```
contract Multiplier {
    //Address of the "promoter" of the contract: she receives fee for each transaction
    address constant private PROMO = 0x5D5fe29339592eEb51c43E54F0a81cA7642B6d2b;
    //Percent received by the "promoter"
    uint constant public PROMO_PERCENT = 7;
    //How many percent for your deposit to be multiplied
    uint constant public MULTIPLIER = 121;

    //The deposit structure holds all the info about the made deposits
    struct Deposit {
        address depositor; //The depositor address
        uint128 deposit;   //The deposit amount
        uint128 expect;    //How much we should pay out (initially it is 121% of deposit)
    }

    Deposit[] private queue;  //The queue of investors
    uint public currentHead = 0; //The index of the first depositor in the queue.

    //This function receives all the deposits stores them  in the queue and make immediate payouts
    function () public payable {
        if(msg.value > 0){
            //Add the investor into the queue. Mark that he expects to receive 121% of deposit back
            queue.push(Deposit(msg.sender, uint128(msg.value), uint128(msg.value*MULTIPLIER/100)));

            //Send some promo to enable this contract to leave long-long time
            uint promo = msg.value*PROMO_PERCENT/100;
            PROMO.send(promo);

            //Pay to first investors in line
            pay();
        }
    }
```

Smart Ponzi

```
contract Multiplier {
    //Address of the "promoter" of the contract: she receives fee for each transaction
    address constant private PROMO = 0x5D5fe29339592eEb51c43E54F0a81cA7642B6d2b;
    //Percent received by the "promoter"
    uint constant public PROMO_PERCENT = 7;
    //How many percent for your deposit to be multiplied
    uint constant public MULTIPLIER = 121;

    //The deposit structure holds all the info about the made deposits
    struct Deposit {
        address depositor; //The depositor address
        uint128 deposit;   //The deposit amount
        uint128 expect;    //How much we should pay out (initially it is 121% of deposit)
    }

    Deposit[] private queue;  //The queue of investors
    uint public currentHead = 0; //The index of the first depositor in the queue.

    //This function receives all the deposits stores them  in the queue and make immediate payouts
    function () public payable {
        if(msg.value > 0){
            //Add the investor into the queue. Mark that he expects to receive 121% of deposit back
            queue.push(Deposit(msg.sender, uint128(msg.value), uint128(msg.value*MULTIPLIER/100)));

            //Send some promo to enable this contract to leave long-long time
            uint promo = msg.value*PROMO_PERCENT/100;
            PROMO.send(promo);

            //Pay to first investors in line
            pay();
        }
    }
```

Smart Ponzi

```
contract Multiplier {
    //Address of the "promoter" of the contract: she receives fee for each transaction
    address constant private PROMO = 0x5D5fe29339592eEb51c43E54F0a81cA7642B6d2b;
    //Percent received by the "promoter"
    uint constant public PROMO_PERCENT = 7;
    //How many percent for your deposit to be multiplied
    uint constant public MULTIPLIER = 121;

    //The deposit structure holds all the info about the made deposits
    struct Deposit {
        address depositor; //The depositor address
        uint128 deposit;   //The deposit amount
        uint128 expect;    //How much we should pay out (initially it is 121% of deposit)
    }

    Deposit[] private queue;  //The queue of investors
    uint public currentHead = 0; //The index of the first depositor in the queue.

    //This function receives all the deposits stores them  in the queue and make immediate payouts
    function () public payable {
        if(msg.value > 0){
            //Add the investor into the queue. Mark that he expects to receive 121% of deposit back
            queue.push(Deposit(msg.sender, uint128(msg.value), uint128(msg.value*MULTIPLIER/100)));

            //Send some promo to enable this contract to leave long-long time
            uint promo = msg.value*PROMO_PERCENT/100;
            PROMO.send(promo);

            //Pay to first investors in line
            pay();
        }
    }
```

Smart Ponzi

```
contract Multiplier {
    //Address of the "promoter" of the contract: she receives fee for each transaction
    address constant private PROMO = 0x5D5fe29339592eEb51c43E54F0a81cA7642B6d2b;
    //Percent received by the "promoter"
    uint constant public PROMO_PERCENT = 7;
    //How many percent for your deposit to be multiplied
    uint constant public MULTIPLIER = 121;

    //The deposit structure holds all the info about the made deposits
    struct Deposit {
        address depositor; //The depositor address
        uint128 deposit;   //The deposit amount
        uint128 expect;    //How much we should pay out (initially it is 121% of deposit)
    }

    Deposit[] private queue;  //The queue of investors
    uint public currentHead = 0; //The index of the first depositor in the queue.

    //This function receives all the deposits stores them  in the queue and make immediate payouts
    function () public payable {
        if(msg.value > 0){
            //Add the investor into the queue. Mark that he expects to receive 121% of deposit back
            queue.push(Deposit(msg.sender, uint128(msg.value), uint128(msg.value*MULTIPLIER/100)));

            //Send some promo to enable this contract to leave long-long time
            uint promo = msg.value*PROMO_PERCENT/100;
            PROMO.send(promo);

            //Pay to first investors in line
            pay();
        }
    }
```

Smart Ponzi

```
contract Multiplier {
    //Address of the "promoter" of the contract: she receives fee for each transaction
    address constant private PROMO = 0x5D5fe29339592eEb51c43E54F0a81cA7642B6d2b;
    //Percent received by the "promoter"
    uint constant public PROMO_PERCENT = 7;
    //How many percent for your deposit to be multiplied
    uint constant public MULTIPLIER = 121;

    //The deposit structure holds all the info about the made deposits
    struct Deposit {
        address depositor; //The depositor address
        uint128 deposit;   //The deposit amount
        uint128 expect;    //How much we should pay out (initially it is 121% of deposit)
    }

    Deposit[] private queue;  //The queue of investors
    uint public currentHead = 0; //The index of the first depositor in the queue.

    //This function receives all the deposits stores them  in the queue and make immediate payouts
    function () public payable {
        if(msg.value > 0){
            //Add the investor into the queue. Mark that he expects to receive 121% of deposit back
            queue.push(Deposit(msg.sender, uint128(msg.value), uint128(msg.value*MULTIPLIER/100)));

            //Send some promo to enable this contract to leave long-long time
            uint promo = msg.value*PROMO_PERCENT/100;
            PROMO.send(promo);

            //Pay to first investors in line
            pay();
        }
    }
```

Smart Ponzi

```
contract Multiplier {
    //Address of the "promoter" of the contract: she receives fee for each transaction
    address constant private PROMO = 0x5D5fe29339592eEb51c43E54F0a81cA7642B6d2b;
    //Percent received by the "promoter"
    uint constant public PROMO_PERCENT = 7;
    //How many percent for your deposit to be multiplied
    uint constant public MULTIPLIER = 121;

    //The deposit structure holds all the info about the made deposits
    struct Deposit {
        address depositor; //The depositor address
        uint128 deposit;   //The deposit amount
        uint128 expect;    //How much we should pay out (initially it is 121% of deposit)
    }

    Deposit[] private queue;  //The queue of investors
    uint public currentHead = 0; //The index of the first depositor in the queue.

    //This function receives all the deposits stores them  in the queue and make immediate payouts
    function () public payable {
        if(msg.value > 0){
            //Add the investor into the queue. Mark that he expects to receive 121% of deposit back
            queue.push(Deposit(msg.sender, uint128(msg.value), uint128(msg.value*MULTIPLIER/100)));

            //Send some promo to enable this contract to leave long-long time
            uint promo = msg.value*PROMO_PERCENT/100;
            PROMO.send(promo);

            //Pay to first investors in line
            pay();
        }
    }
```

Smart Ponzi

```
contract Multiplier {
    //Address of the "promoter" of the contract: she receives fee for each transaction
    address constant private PROMO = 0x5D5fe29339592eEb51c43E54F0a81cA7642B6d2b;
    //Percent received by the "promoter"
    uint constant public PROMO_PERCENT = 7;
    //How many percent for your deposit to be multiplied
    uint constant public MULTIPLIER = 121;

    //The deposit structure holds all the info about the made deposits
    struct Deposit {
        address depositor; //The depositor address
        uint128 deposit;   //The deposit amount
        uint128 expect;    //How much we should pay out (initially it is 121% of deposit)
    }

    Deposit[] private queue;  //The queue of investors
    uint public currentHead = 0; //The index of the first depositor in the queue.

    //This function receives all the deposits stores them  in the queue and make immediate payouts
    function () public payable {
        if(msg.value > 0){
            //Add the investor into the queue. Mark that he expects to receive 121% of deposit back
            queue.push(Deposit(msg.sender, uint128(msg.value), uint128(msg.value*MULTIPLIER/100)));

            //Send some promo to enable this contract to leave long-long time
            uint promo = msg.value*PROMO_PERCENT/100;
            PROMO.send(promo);

            //Pay to first investors in line
            pay();
        }
    }
```

Smart Ponzi

```
contract Multiplier {
    //Address of the "promoter" of the contract: she receives fee for each transaction
    address constant private PROMO = 0x5D5fe29339592eEb51c43E54F0a81cA7642B6d2b;
    //Percent received by the "promoter"
    uint constant public PROMO_PERCENT = 7;
    //How many percent for your deposit to be multiplied
    uint constant public MULTIPLIER = 121;

    //The deposit structure holds all the info about the made deposits
    struct Deposit {
        address depositor; //The depositor address
        uint128 deposit;   //The deposit amount
        uint128 expect;    //How much we should pay out (initially it is 121% of deposit)
    }

    Deposit[] private queue;  //The queue of investors
    uint public currentHead = 0; //The index of the first depositor in the queue.

    //This function receives all the deposits stores them  in the queue and make immediate payouts
    function () public payable {
        if(msg.value > 0){
            //Add the investor into the queue. Mark that he expects to receive 121% of deposit back
            queue.push(Deposit(msg.sender, uint128(msg.value), uint128(msg.value*MULTIPLIER/100)));

            //Send some promo to enable this contract to leave long-long time
            uint promo = msg.value*PROMO_PERCENT/100;
            PROMO.send(promo);

            //Pay to first investors in line
            pay();
        }
    }
```

Smart Ponzi

```
//Used to distribute all the money on contract to the first investors starting from the head of queue
function pay() private {
    // The current balance of the contract
    uint128 money = uint128(address(this).balance);

    //Cycle on the queue
    for(uint i=0; i<queue.length; i++){

        uint idx = currentHead + i;  //the index of the currently first investor

        Deposit storage dep = queue[idx]; //the info of the first investor

        if(money >= dep.expect){  //If we have enough money on the contract to fully pay to investor
            dep.depositor.send(dep.expect); //Send money to him
            money -= dep.expect;            //update money left

            //the investor is fully paid and she is removed from the queue
            delete queue[idx];
        }else{
            //No enough money, so partially pay to investor
            dep.depositor.send(money); //Send to her the money left
            dep.expect -= money;       //Update her expected amount
            break;
        }
    }

    currentHead += i; //Update the index of the current first investor
}
```

Smart Ponzi

```
//Used to distribute all the money on contract to the first investors starting from the head of queue
function pay() private {
    // The current balance of the contract
    uint128 money = uint128(address(this).balance);

    //Cycle on the queue
    for(uint i=0; i<queue.length; i++){

        uint idx = currentHead + i;  //the index of the currently first investor

        Deposit storage dep = queue[idx]; //the info of the first investor

        if(money >= dep.expect){  //If we have enough money on the contract to fully pay to investor
            dep.depositor.send(dep.expect); //Send money to him
            money -= dep.expect;            //update money left

            //the investor is fully paid and she is removed from the queue
            delete queue[idx];
        }else{
            //No enough money, so partially pay to investor
            dep.depositor.send(money); //Send to her the money left
            dep.expect -= money;       //Update her expected amount
            break;
        }
    }

    currentHead += i; //Update the index of the current first investor
}
```

Smart Ponzi

```
//Used to distribute all the money on contract to the first investors starting from the head of queue
function pay() private {
    // The current balance of the contract
    uint128 money = uint128(address(this).balance);

    //Cycle on the queue
    for(uint i=0; i<queue.length; i++){

        uint idx = currentHead + i;  //the index of the currently first investor

        Deposit storage dep = queue[idx]; //the info of the first investor

        if(money >= dep.expect){  //If we have enough money on the contract to fully pay to investor
            dep.depositor.send(dep.expect); //Send money to him
            money -= dep.expect;            //update money left

            //the investor is fully paid and she is removed from the queue
            delete queue[idx];
        }else{
            //No enough money, so partially pay to investor
            dep.depositor.send(money); //Send to her the money left
            dep.expect -= money;       //Update her expected amount
            break;
        }
    }

    currentHead += i; //Update the index of the current first investor
}
```

Smart Ponzi

```
//Used to distribute all the money on contract to the first investors starting from the head of queue
function pay() private {
    // The current balance of the contract
    uint128 money = uint128(address(this).balance);

    //Cycle on the queue
    for(uint i=0; i<queue.length; i++){

        uint idx = currentHead + i;  //the index of the currently first investor

        Deposit storage dep = queue[idx]; //the info of the first investor

        if(money >= dep.expect){  //If we have enough money on the contract to fully pay to investor
            dep.depositor.send(dep.expect); //Send money to him
            money -= dep.expect;            //update money left

            //the investor is fully paid and she is removed from the queue
            delete queue[idx];
        }else{
            //No enough money, so partially pay to investor
            dep.depositor.send(money); //Send to her the money left
            dep.expect -= money;       //Update her expected amount
            break;
        }
    }

    currentHead += i; //Update the index of the current first investor
}
```

IMT SCHOOL FOR ADVANCED STUDIES LUCCA

Some features