Pycoin: A Decentralized Platform for Secure Smart Contracts

Version: 0.01
Cohomology Labs
January 18, 2017

Abstract.

Pycoin is a decentralized application for writing, testing and deploying mathematically verifiable secure smart contracts in python language that behave exactly as intended.

Unlike Ethereum smart contracts with a history of repeated hacks and stolen funds, hundreds and millions of dollars, Pycoin ensures that no run time errors or intentional/unintentional bugs can be used as exploits to steal funds and corrupt Pycoin smart contracts.

Pycoin uses Simplicity, a new typed, combinator-based, functional language without loops and recursion, designed to be used for crypto-currencies and blockchain applications as its target language and Python, a highly popular and versatile high-level programming language as its source. Simplicity was released recently by Blockstream, an influential decentralized application development company.

Owing to its Turing incompleteness, Simplicity is amenable to static analysis that can be used to derive upper bounds on the computational resources needed, prior to execution. While Turing incomplete, Simplicity can express any finitary function, which we trust is enough to build useful “smart contracts” for blockchain and blockchain-free decentralized applications.

This work:

(a) Introduces the Pycoin which binds to DexOS, a real-time unforkable blockchain and DAG-based blockchain-free decentralized network of XRB (Raiblocks) family, gives an overview of the protocol, and walks through several components in detail.

(b) Formalizes decentralized Secure Smart Contract (SSC) structures and their properties, then constructs Pycoin as a Decentralized Secure Smart Contract Platform.

(c) Discusses use cases, connections to other systems, and how to use the protocol.

Note: Pycoin is a work in progress. Active research is under way, and new versions of this paper will appear at pycoin.io. For comments and suggestions, contact us at info@pycoin.io or send us an encrypted P2P message at our Bitmessage address:

BM-NBQe5VrbE79VUB9j7djWCTin2N42GJ2i

Contents

1 Introduction

DApp basically stands for decentralized applications. The first DApp was in fact the bitcoin itself. And these blockchains and other decentralized open ledger technologies relies on smart contracts i.e. predefined protocols that allow large, ad-hoc, groups of users to transfer value between themselves without needing to trust each other or any central authority. Bitcoin uses bitcoin script to create such smart contracts and Ethereum uses EVM with solidity to create such smart contracts. Both bitcoin script and EVM are much different from each other. Bitcoin script is Turing incomplete thus less complex but less powerful although enough powerful to create a reliable smart contract whereas the EVM is a Turing complete thus more complex and more powerful to create more secure smart contracts if only used well. Meanwhile Ethereum is one of the leading open software platform based on blockchain technology that enables developers to build and deploy decentralized applications.

Ethereum can also be used to build Decentralized Autonomous Organizations (DAO). A DAO is fully autonomous, decentralized organization with no single leader. DAO's are run by programming code, on a collection of smart contracts written on the Ethereum blockchain. The code is designed to replace the rules and structure of a traditional organization, eliminating the need for people and centralized control. A DAO is owned by everyone who purchases tokens, but instead of each token equating to equity shares & ownership, tokens act as contributions that give people voting rights.
But in 2016 'The DAO' project got hacked and since then Ethereum blockchain was forced split itself and their so called smart contracts are less trusted.

Our goal is to create a platform that enables which can be more reliable and easy to understand and create smart contracts. And Pycoin is a decentralized application for writing, testing and deploying mathematically verifiable secure smart contracts(SSCs) in python language that behave exactly as intended.

1.1 Elementary components

There are noticeable common features of DApps:

- **Open Source**. Ideally, it should be governed by autonomy and all changes must be decided by the consensus, or a majority, of its users. Its code base should be available for scrutiny.
- **Decentralized**. All records of the application's operation must be stored on a public and decentralized blockchain to avoid pitfalls of centralization.
- **Incentivized**. Validators of the blockchain should be incentivized by rewarding them accordingly with cryptographic tokens.
- **Protocol.** The application community must agree on a cryptographic algorithm to show proof of value. For example, Bitcoin uses Proof of Work (pow) and Ethereum is currently using PoW with plans for a hybrid PoW/Proof of Stake (PoS) in the future.

Pycoin binds to DexOs which is an advanced minerless unforkable blockchain inspired by Algorand, based on OrchOr. Pycoin can also be deployed on DAG based, blockchainfree networks like XRB (Raiblocks) family.

Pycoin uses Simplicity, a new typed, combinator-based, functional language without loops and recursion, designed to be used for crypto-currencies and blockchain applications as its target language and Python, a highly popular and versatile high-level programming language as its source. Simplicity was released recently by Blockstream, an influential decentralized application development company.

Owing to its Turing Incompleteness, Simplicity is amenable to static analysis that can be used to derive upper bounds on the computational resources needed, prior to execution. While Turing incomplete, Simplicity can express any finitary function, which we believe is enough to build useful “smart contracts” for blockchain and blockchainfree decentralized applications(DApps).

1.2 Protocol overview

Instead of bulky blocks used as transaction containers in existing blockchain designs, the DAG builds a graph of transactions which reference older transactions and thus can confirm transactions immediately when they are received by a node instead of having to wait for the next block. This means that every node consists of multiple layers of transactions. When a transaction is registered in a node, it first has to verify two other transactions before his transaction will be verified. Those two transactions are chosen according to an algorithm. The node has to check if the two transactions are not conflicting. For a node to issue a valid transaction, it must solve a cryptographic puzzle similar to those in the Bitcoin network (Proof of Work).

1.3 Paper Organization

In the report basically we are talking about Pycoin. Starting from the basic definitions of smart contracts and how they can be used in a decentralized network, to what are the other platforms to create smart contracts like Ethereum. Then we talked about the basic component to write smart contracts which is a language and the basic property of the language which is Turing completeness.
While Ethereum uses Turing completeness language to create its smart contracts Pycoin uses Turing incompleteness language to create its smart contracts. Then we moved to the properties possessed by the smart contracts, and securities involved in smart contracts and its management.

In the next section we are talking about why we need another decentralized application if we have a Turing complete language based Ethereum. Then it moves to forkability and how can we avoid it by implementing an Algorand-like real-time unforkable blockchain DexOS. Next section is about running Pycoin-based secure smart contracts on DAG-based blockchain free networks. There are also the uses cases of Pycoin in real life.

The next topic in our report is Pycoin, its data structure and the protocol. Then what are the features Pycoin offers and what are the other requirements which make the Pycoin a complete DApp Platform that can enable other DApps and smart contracts on both blockchainfree networks like XRB/Raiblocks capable of performing 7000 transactions per second and realtime unforkable blockchains like Algorand and DexOS.

2 Definition of a Secure Smart Contract

Smart contracts [1] help you exchange money, property, shares, or anything of value in a transparent, conflict-free way while avoiding the services of a middleman.

Basically it is a computer program that directly controls the transfer of values between the network without needing to trust each other or any central authority .

The best way to describe smart contracts is to compare the technology to a vending machine. Ordinarily, you would go to a lawyer or a notary, pay them, and wait while you get the document. With smart contracts, you simply drop a Bitcoin into the vending machine (i.e. ledger), and your escrow, driver's license, or whatever drops into your account. More so, smart contracts not only define the rules and penalties around an agreement in the same way that a traditional contract does, but also automatically enforce those obligations.

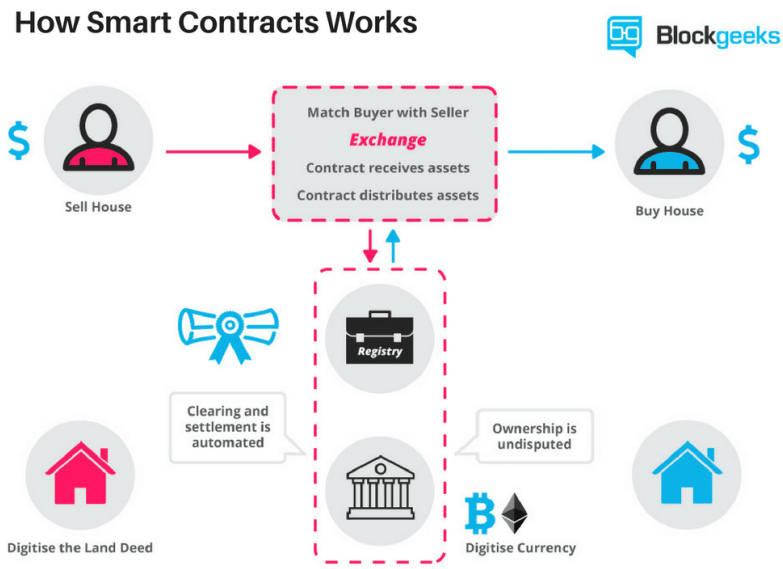
Smart contracts, meanwhile, work hand-in-hand with block chain technology and have the potential to automate and also disrupt processes in many industries.

Whereas a traditional legal contract defines the rules around an agreement between multiple people or parties, smart contracts go a step further and actually enforce those rules by controlling the transfer of currency or assets under specific conditions. In a smart contract approach, an asset or currency is transferred into a program and the program runs this code and at some point it automatically validates a condition and it automatically determines whether the asset should go to one person or back to the other person, or whether it should be immediately refunded to the person who sent it.

A smart contract not only defines the rules and penalties around an agreement in the same way that a traditional contract does, but it can also automatically enforce those obligations.

Smart contracts guarantee a very, very specific set of outcomes. There's never any confusion and there's never any need for litigation. It's simply a very limited, computer-guaranteed set of outcomes.

This is how smart contracts works



3.1 Turing Incompleteness

Every decentralized application needs a language to write smart contracts. The bitcoin uses bitcoin script to write its smart contract. Bitcoin script is a Turing incomplete language. Whereas Ethereum uses EVM (which is a Turing complete language) and solidity to write its smart contracts. Pycoin uses simplicity as its core language to write its smart contracts. Simplicity is a Turing incomplete language.

A Turing-complete system is called Turing equivalent if every function it can compute is also Turing computable; i.e., it computes precisely the same class of functions as do Turing machines. Alternatively, a Turing-equivalent system is one that can simulate, and be simulated by, a universal Turing machine.

There are pros and cons of a language being Turing completeness. A Turing completeness language is more expressive having loops and jump statements which also make it difficult to use and the programs written in Turing completeness language are more complex. Another disadvantage of having a Turing completeness language is, it is not possible to calculate computational effort before the execution of the program and you need to put a limiter to avoid infinite loops in the program like gas in Ethereum. If a programming language doesn't have any jump or looping statements, then that language is called Turing incompleteness. A Turing incompleteness language's program is amenable to static analysis i.e. an upper bound for that program can be calculated before its execution and allows to place the limits on the amount of computation a transaction can have.

3.2 Properties

The key properties of smart contracts are:

- Autonomy
- Decentralization
- Auto sufficiency
- Safety
- Precision

Autonomy implies that after a smart contact launches, the deal initiator does not have to participate any more in the process. Smart contracts are not focused on one central server but are distributed by various network points so they can be referred to as being *decentralized*. *Auto-sufficiency* supposes that contracts are able to collect money, realize transactions, distribute resources, issue and spend funds to allow a larger capacity of storage and computation power.

Safety of the data is maintained, since the data in the decentralized registry cannot be lost or modified by the use of cryptography. Precision means that no mistakes can be made due to the absence of hand filled forms.

3.3 Security and risk management

A smart contract is “a computerized transaction protocol that executes the terms of a contract”. A block chain-based smart contract is visible to all users of said block chain. However, this leads to a situation where bugs, including security holes, are visible to all yet may not be quickly fixed.

Such an attack, difficult to fix quickly, was successfully executed on The DAO in June 2016, draining US$50 million in Ether while developers attempted to come to a solution that would gain consensus. The DAO program had a time delay in place before the hacker could remove the funds, a hard fork of the Ethereum software was done to claw back the funds from the attacker before the time limit expired.

Issues in Ethereum smart contracts in particular include ambiguities and easy-but-insecure constructs in its contract language Solidity, compiler bugs, Ethereum Virtual Machine bugs, attacks on the block chain network, the immutability of bugs and that there is no central source documenting known vulnerabilities, attacks and problematic constructs.

4. Computational Boundedness and Decidable Logic

4.1 Motivation

Preventing loss of computation

Since in Ethereum, the EVM is Turing complete language, so it’s not feasible to static analysis, i.e. we cannot calculate the upper bound on the calculations program will make before its execution, so each program must be provided with gas (which is paid for in Ethereum’s unit of account, ether, to the miner of block containing the transaction) to place the counter and avoid infinite loops. As the program runs out of the gas, the transaction is nullified but the gas is still paid to the miner to ensure they are compensated for the computational efforts they made.

With Turing incomplete programming language like simplicity, static analysis allows the protocol to place limits on the amount of computation a transaction can have, so that nodes running the protocol are not overly burdened. Furthermore, the static analysis can provide program creators with a general-purpose tool for verifying that the programs they build will always fit within these limits. Additionally, it is easy for the other participants in a contract to check the bounds on smart contract's programs themselves.

Testing of program

They may be buggy, just like any other code. Debugging and testing them is quite involved due to the lack of tools. If Smart Contracts take off in popularity, expect new services and technology focused on doing all types of security checks before they are deployed.

Pycoin uses simplicity which has formal semantics. And thus Formal semantics that work with proof-assistant software provide the opportunity for contract developers to reason about their programs to rule out logical errors and to help avoid scenarios like the DAO and Parity's multi-

signature program failure. Pycoin employs an interpolation of popular programming language Python as its source language codenamed PyLegal.

4.2 Realtime Unforkable Blockchain of Pycoin-DexOS

Pycoin runs on its sister platform DexOS which is being engineered as a real-time unforkable blockchain besides being also compatible with blockchainfree DAG and lattice-based decentralized networks like XRB(Raiblocks). Since XRB is an existing running DAG-based cryptocurrency network, Pycoin secure smart contracts can be immediately deployed on a fork of XRB.

Blockchain systems are distributed implementations of a chain of blocks. Each node can issue a cryptographically signed transaction to transfer digital assets to another node or can create a new block of transactions, and append this block to its current view of the chain. Due to the distributed nature of this task, multiple nodes may append distinct blocks at the same index of the chain before learning about the presence of other blocks, hence leading to a forked chain or a tree. For nodes to eventually agree on a unique state of the system, nodes apply a common strategy that selects a unique branch of blocks in this tree. That's how they avoid forking. There are so many mechanisms to make the blockchain unforkable like the byzantine agreement protocol and many other schemes.

The Unforkable Realtime Blockchain DexOS showcases unparalleled advantages over the blockchains currently in use in cryptocurrency and business settings.

DexOS is based on Effortless One-by-One Byzantine Agreement

There are several techniques to manage blockchain-based ledgers have been proposed:

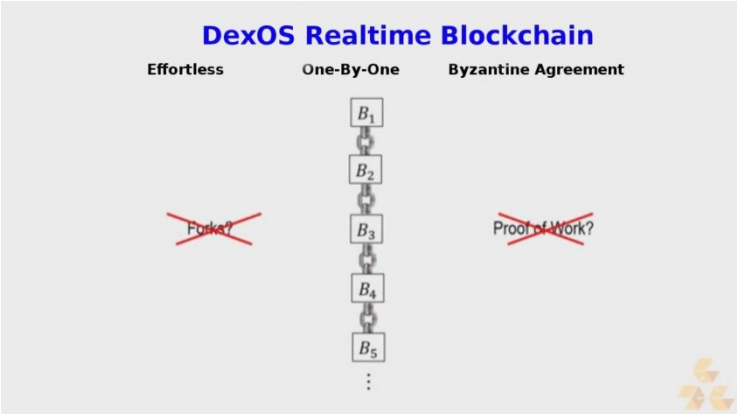
1. Proof-of-Work (PoW)

2. Proof-of-Stake (PoS)

3. Practical Byzantine fault-tolerance or some combination.

Currently, however, ledgers can be inefficient to manage. For example, Bitcoin's proof-of-work(PoW) approach requires a vast amount of computation, is wasteful and scales poorly. In addition, it de facto concentrates power in very few hands because of miners' requirements.

DexOS is inspired by Algorand. The design goal of Algorand was building a new method to implement a public ledger that offers the convenience and efficiency of a centralized system run by a trusted and inviolable authority, without the inefficiencies and weaknesses of current decentralized implementations.

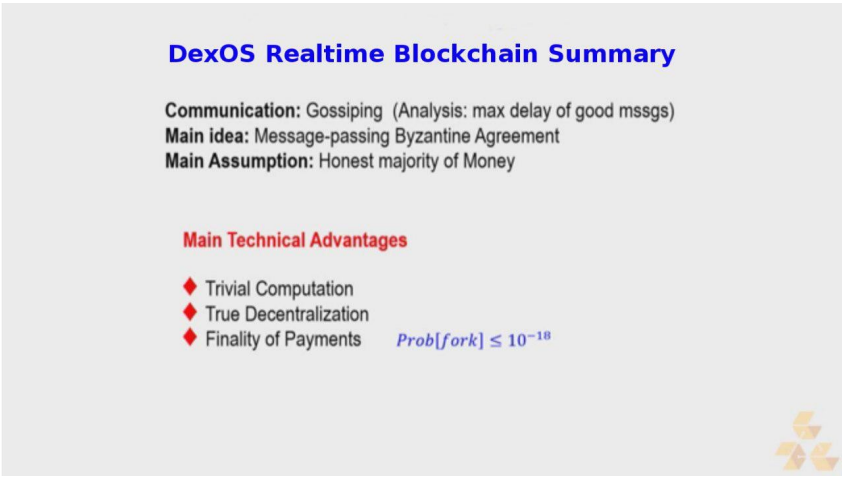


For DexOS we use algorithmic randomness to select, based on the ledger constructed so far, a set of verifiers who are in charge of constructing the next block of valid transactions. Naturally, we ensure that such selections are provably immune from manipulations and unpredictable until the last minute, but also that they ultimately are universally clear. DexOS's approach is quite democratic, in the sense that neither in principle nor de facto it creates different classes of users (as "miners" and "ordinary users" in Bitcoin). In DexOS "all power resides with the set of all users".

One notable property of DexOS is that its transaction history may fork only with very small probability (e.g., one in a trillion, that is, or even 10^-18).every underlying unconventional mechanism of DexOS based on a single linear Blockchain makes sure there are no multiple blockchains and that's why forking is not possible in the absence of PoW(proof of work) miners.

Here users verify each other's' transactions and there are no special class of miners. DexOS once fully deployed with minimum 10,000 plus nodes will be capable of processing 2 million transactions per second(tps).

These speeds are unparalleled, leaving the closest competitors far behind. In comparison, Visa's network has a peak capacity of around 56,000 transactions per second and the Bitcoin network is limited to around seven transactions per second.



DexOS also differs from existing blockchains as it is designed never to fork. Transactions are verified before they are recorded on the blockchain. "The DexOS Blockchain is unforkable. Mainstream traditional blockchains do not order all blocks of transactions with each other. Instead, they tolerate that multiple blocks be appended at the same index of the blockchain, hence causing what are known as forks. To prevent an attacker exploiting these forks to double spend in two of

these blocks, these blockchains try to trim out these forks. Instead of curing of the forks, DexOS prevents them: the consensus among the participants occurs before a new block is added so that the blockchain remains a chain.

DexOS is capable of processing millions of transactions per second coming from a potentially unbounded number of clients. It offers a performance that scales horizontally, which ensures the security of transactions.

Compared to traditional blockchains which becomes slower and slower as network becomes crowded with more users and transactions, DexOS becomes faster and faster as more users join the network.

4.3 Use Cases of Pycoin

Pycoin has got a number of real world uses. Primarily it can easily replace less-secure Ethereum smart contracts written in Turing Complete Solidity language. Being Turing Incomplete Pycoin is many times more secure than Ethereum Contracts. In terms productivity, since Pycoin is based on an interpolation of Python language known as PyLegal, Pycoin will be more appealing to more than 4 million Python developers worldwide. It would cut the learning curve short for creating secure smart contracts in PyLegal.

Tokenized Financial Assets and Derivatives Contracts: Pycoin can tokenize all global/local remittances, payments, trading and hedging. Pycoin can also be used to create automated and decentralized financial instruments - such as bonds, commodities, debt instruments and securities denominated in both cryptocurrencies and fiat currencies.

Insurance companies will be able to provide temporal liability insurance. For instance, using Pycoin, an insurance company could charge rates differently based on where and under what conditions customers are operating their vehicles. A car driven on a clear day (ascertained by gathering information about the weather conditions from a weather service), in an area where all the roads are repaired (verified with information about road repairs supplied by the Department of Motor Vehicles, for instance), would be charged a lower rate compared with a car that's being operated in bad weather, perhaps on pothole-filled roads.

Real estate -Instead of transferring payment for the property through an escrow account, buyers would send payment to a ledger in a blockchain system.

The blockchain system has encoded in that value the conditions by which that value may be further spent. The buyer and the seller sign electronically, digitally, this contract, the value is transferred, and the ownership of that residential property is transferred. In the case of a dispute, a real estate attorney would act as an arbiter and as arbiter would transfer value to the party he sides with.

Supply chain -Pycoin will also be useful in the supply chain. One can execute contracts that say, If I receive cash on delivery at this location in a developing, emerging market, then this other product, many, many links up the supply chain, will trigger a supplier creating a new item since the existing item was just delivered in that developing market.

4 Pycoin for DAG-based decentralized networks: data structures and protocol

4.1 Setting

Pycoin uses simplicity as core language to do the computations in the smart contracts. Simplicity cannot express general computation. It can only express finitary functions, because each Simplicity type contains only finitely many values. However, within this domain, Simplicity's set of combinators is complete: any function between Simplicity's types can be expressed which is sufficient to program a smart contract.

The core of Simplicity consists of nine combinators for building expressions. They capture the fundamental operations for the three flavors of types in Simplicity. The typing rules for these nine combinators.

We can classify the combinators based on the flavor of types they support.

- The unit term returns the singular value of the unit type and ignores its argument.
- The injl and injr combinators create tagged values, while the case combinator, Simplicity's branching operation, evaluates one of its two sub-expressions based on the tag of the first component of its input.
- The pair combinator creates pairs, while the take and drop combinators access first and second components of a pair respectively.
- The iden and comp combinators are not specific to any flavor of type.

The iden term represents the identity function for any type and the comp combinator provides function composition.

Simplicity expressions form an abstract syntax tree. The leaves of this tree are either iden or unit terms. The nodes are one of the other seven combinators.
Each node has one or two children depending on which combinator the node represents.

4.2 Data structures

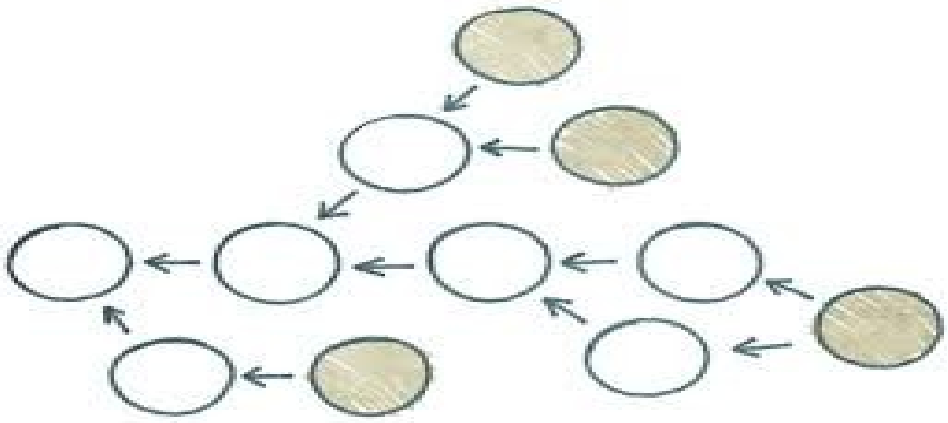
Pycoin uses initially a directed acyclic graph decentralized network for deploying on a fork of DAG-based XRB(RaiBlocks). DAGs more reliable and efficient than traditional blockchains.

DAG (directed acyclic graph) – is a finite directed graph with no directed cycles. That is, it consists of finitely many vertices and edges, with each edge directed from one vertex to another, such that there is no way to start at any vertex *v* and follow a consistently-directed sequence of edges that eventually loops back to *v* again. Equivalently, a DAG is a directed graph that has a topological ordering, a sequence of the vertices such that every edge is directed from earlier to later in the sequence.

Mathematical properties of DAG

- Reachability: The reachability relationship in any directed acyclic graph can be formalized as a partial order ≤ on the vertices of the DAG. In this partial order, two vertices *u* and *v* are ordered as $u \leq v$ exactly when there exists a directed path from *u* to *v*in the DAG; that is, when *v* is reachable from *u*.
- Transitive closure: If *G* is a DAG, its transitive closure is the graph with the most edges that represents the same reachability relation. It has an edge $u \to v$ whenever *u* can reach *v*. That is, it has an edge for every related pair $u \leq v$ of distinct elements in the reachability relation of *G*, and may therefore be thought of as a direct translation of the reachability relation ≤ into graph-theoretic terms.
- Transitive reduction: The transitive reduction of a DAG *G* is the graph with the fewest edges that represents the same reachability relation as *G*. It is a subgraph of *G*, formed by discarding the edges $u \to v$ for which *G* also contains a longer path connecting the same two vertices. Like the transitive closure, the transitive reduction is uniquely defined for DAGs. In contrast, for a directed graph that is not acyclic, there can be more than one minimal subgraph with the same reachability relation.
- Topological ordering: Every directed acyclic graph has a topological ordering, an ordering of the vertices such that the starting endpoint of every edge occurs earlier in the ordering than the ending endpoint of the edge. The existence of such an ordering can be used to characterize DAGs: a directed graph is a DAG if and only if it has a topological ordering. In general, this ordering is not unique; a DAG has a unique topological ordering if and only if it has a directed path containing all the vertices, in which case the ordering is the same as the order in which the vertices appear in the path.

Directed acyclic graph



4.3 Protocol

Raiblocks, Dagcoins/Byteball and Iota are some of the major DAG based cryptocurrencies. They all have different consensus protocol. Like Byteball achieves consensus by relying on a “main-chain” comprised of honest, reputable and user-trusted “witnesses”, while IOTA achieves consensus via the cumulative PoW of stacked transactions. RaiBlocks achieves consensus via a balance-weighted vote on conflicting transactions.

In DAG-based cryptocurrencies each new transaction confirms one or more previous transactions. As a result, transactions perform a structure which represents a directed graph with no directed cycles. The major problem of developing DAG-based cryptocurrencies was limitations on width growth. The situation, when users can pick the same transaction as a parent, was forbidden. Limitations on choosing old transactions as parent transactions also caused problems. Thus, the key objective of designing a new cryptocurrency was developing an algorithm, which during operation presented transactions in the form of a DAG-chain, a directed acyclic graph, whose length is much more than its width (Figure 1). In this case, the graph’s length corresponds to the number of edges of the longest chain, which connects the first transaction with the current transaction. The graph’s width is the number of points, having the same parent point (transaction) and not connected by edges.

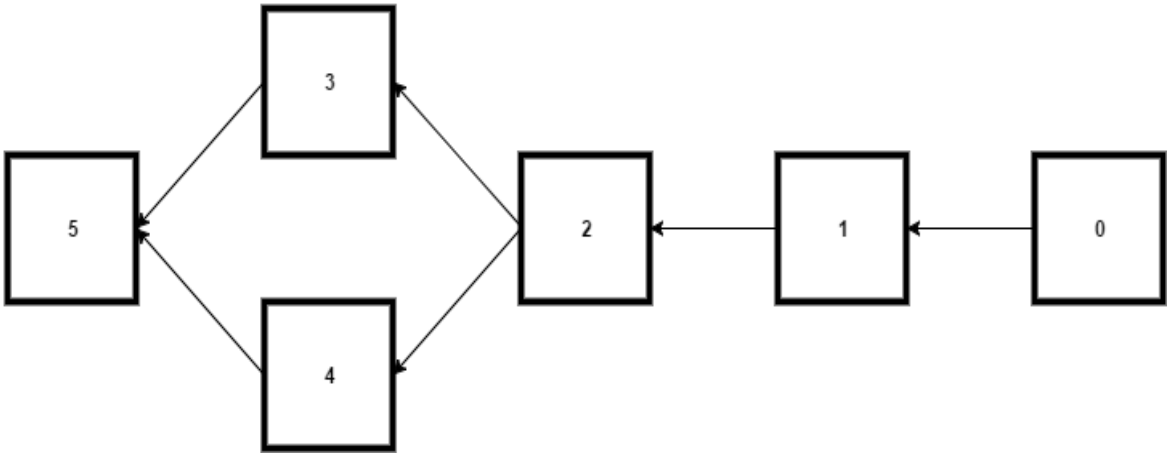


Figure 1: A graphic representation of a DagCoin cryptocurrency. Transaction 2 approves Transactions 3 and 4, which, in their turn, approve Transaction 5, because within this cryptocurrency a new transaction can approve one or more transactions.

All the transaction types in the network must have a work field which allows the miner to compute A nonce such that the hash of the nonce concatenated with the previous field in receive/change/send transactions or the account field in an open transaction below a certain field. Unlike bitcoin, the proof of work in this is similar to hash cash and can be computed on the order of seconds[2]. For the transaction which is sent, the PoW can be precomputed using the previous block. This will make the transactions to appear instantaneously at the end of blockchain.

To validate a transaction [3] In a DAG based crypto currency the block must have the following attribute:

1) The block must not already be in the ledger (duplicate transaction).
2) Must be signed by the account's owner.
3) The previous block is the head block of the account chain. If it exists but is not the head, it is a fork.
4) The account must have an open block.
5) The computed hash meets the PoW threshold requirement.

If it is a receive block, check if the source block hash is pending, meaning it has not already been redeemed. If it is a send block, the balance must be less than the previous balance.

We know that Pycoin uses simplicity as its core language to create smart contracts, and simplicity is a Turing incomplete language, so it is amenable to static analysis. Using static analysis, we can quickly compute an upper bound on these sorts of resource costs.

These kinds of static analyses are simple recursive functions of Simplicity expressions, and the intermediate results for sub-expressions can be shared. By using a DAG for the in-memory representation of Simplicity expressions, we can transparently cache these intermediate results. This means the time needed to compute static analysis is proportional to the size of the DAG representing the Simplicity expression, as opposed to the time needed for dynamic analysis such as evaluation, which may take time proportional to the size of the tree representing the Simplicity expression.

4.4 Guarantees and requirements

Increasing speed

Low speed is one of the most wide-spread problems with traditional Blockchain-based cryptocurrencies. It is due to the fact that it takes too long to process a block of transactions, which is the primary element of the operation of any cryptocurrency and represents a large data structure. This issue leads to mining becoming gainless for individual miners with low technological capabilities, as mining pools (with higher speed) can gain more benefit in the mining process. Blockchain-free cryptocurrencies, in their turn, outrun Blockchain-based ones, because the primary element of their operation is a single transaction, which takes much less time to process both in a mining pool and with an individual miner.

Increasing reliability

The mechanism of multiple confirmations of transactions in DAG-based cryptocurrencies makes it possible to increase security of payments and decrease the possibility of "double expenses", which is a situation when the chain of transactions breaks into branches and every branch "doesn't know" about the expenditures and transfers going on within the other branch. Theoretically, it gives an opportunity to spend the cryptocurrency twice. Blockchain-free cryptocurrencies overcome this problem by means of direct and indirect confirmations, whose descriptions are unique for every currency. For example, in IOTA a situation when there are at least to edges of confirmation

between two transactions is considered an indirect confirmation. Consequently, one edge of confirmation is viewed as a direct confirmation.

Simplifying mining

In order to fully understand this principle of Blockchain-free cryptocurrencies, one should describe mining process within traditional Blockchain-based cryptocurrencies. Mining is carried out by means of calculating the *nonce* component of a block of transactions. The difficulty of this calculation is defined beforehand and is based upon the non-invertibility of one-way hash-function. Besides, mining has a competitive nature: one who is the first to calculate *nonce*, which fulfills the condition, will get the reward. For instance, with Bitcoin the reward is 50 BTC. Presently, when we have such widespread cryptocurrencies as Bitcoin, individual mining becomes unprofitable. Having lower technological possibilities, individual miners have little chance to calculate the value fast and get the reward. On the other hand, mining pools, getting together many miners under one pool, become more profitable. The block is calculated by all the miners inside the pool: in case of a win, the charges of the pool are subtracted from the overall reward and the sum left is divided between all the other participants. Such a method reduces the profit for every individual miner, but raises the chance for success. Such a phenomenon is called the mining pools oligopoly. When developing Blockchain-free cryptocurrency developers try to avoid this phenomenon. It is achieved by means of designing such algorithms of operation, which do not give a clear technological advantage when mining. Particularly, this is achieved by means of speeding up the processing of transactions, which is connected with the key role of transactions and not blocks in Blockchain-free cryptocurrencies.

The requirements for the Pycoin are:

- Since Pycoin is a DApp which can be used to writing and testing secure smart contracts, a language is needed to program the smart contracts. In Pycoin simplicity is the main choice for the language as it is Turing incomplete, so amenable to static analysis and the calculation effort can be calculated prior to its execution. Also simplicity's formal semantics can be a good choice for the testing of programs so that buggy code can be avoided in smart contracts.

- Pycoin needs a decentralized network to act upon. Here DAG based decentralized network is chosen. As it is already shown above why DAG based network is better than traditional forkable blockchains.

- For implemented DAG based network a well-defined PoW protocol is needed to validate a block which basically contains a single transaction.

- Pycoin uses the python language (actually PyLegal, a Turing Incomplete interpolation of Python with Simplicity as the target language) for the development of secure smart contracts-SSCs.

5.1 Integration with other systems

One system/crypto currencies can integrate with other system/crypto currencies especially those based on PoW/PoS blockchains via use of sidechains and drivechains e.g. Bitcoin and Ethereum. Basically sidechains provide a mechanism to transfer one system's currency to another system's currency securely. Also, a sidechain is a two way peg mechanism. That is an account holder in one blockchain system say bitcoin can send bitcoins to another blockchain system say Ethereum and the mechanism of a sidechain will make sure that the bitcoins from bitcoin blockchain is converted to ethers based on the real-time conversion rates and then moved into the Ethereum's blockchain system. As sidechains are two way peg system so we also can send ethers to bitcoin blockchains. Although sidechains are running projects, there has been found so many flaws and security issues and mining problems.

Due to the problems and securities issues involved in the sidechains, Pycoin has decided to use Hashrate Escrows and Drivechains pioneered by Paul Sztorc in order to move value from Bitcoin and other forks of Bitcoin e.g. Bitcoin Cash and Bitcoin Gold to Pycoin. We plan to merge a drivechain into Bitcoin Core, then we will insert a few parts of our sidechain template and insert a few parts into Algorand-inspired DexOS — the code for deposits (crediting Pycoin BTC for main-to-side transfers) and withdrawals (aggregating side-to-main transfers and broadcasting an easily-visible WT^). Same mechanism will be repeated for other forks of Bitcoin such as Bitcoin Cash and Bitcoin Gold.

This will help us move almost a quarter trillion dollar and growing value from BTC, BCH and BTC Gold to Pycoin and DexOS.

5.2 Future Work

- Pycoin uses Simplicity as its target language which has no unbounded loops or recursion. It is possible to build smart contracts with state carried through loops using covenants without requiring unbounded loops within Simplicity itself. Bounded loops, such as the 64 rounds needed by our SHA-256 implementation, can be achieved by unrolling the loop. Because of sub-expression sharing, this doesn't unreasonably impact program size. We do not directly write Simplicity, rather we use functions written in Coq or Haskell to generate Simplicity. These languages do support recursion and we use loops in these meta-languages to generate unrolled loops in Simplicity.

- Simplicity has no function types and therefore no higher-order functions. While it is possible to compute upper bounds of computation resources of expressions in the presence of function types, it likely that those bounds would be so far above their actual needs that such analysis would not be useful.

- **Incentives** In order to encourage Pycoin users to participate, i.e., be online when selected and pay the network cost of operating Pycoin, the system may need to include incentives, possibly in form of a reward mechanism. Designing and analyzing an incentive mechanism includes many challenges, such as ensuring that users do not have perverse incentives

(e.g., to withhold votes), and that malicious users cannot “game the system” to obtain more rewards than users who follow the protocol.

- The design and implementation of PyLegal, a Turing Incomplete interpolation of Python language is undergoing.

5.3 References:

1) https://blockgeeks.com/guides/smart-contracts/
2) http://www.hashcash.org/papers/hashcash.pdf
3) https://raiblocks.net/media/RaiBlocks_Whitepaper__English.pdf
4) Tauchain http://tauchain.org/
5) Hash Rate Escrows-Drivechains
   https://github.com/drivechain-project/docs/blob/master/bip1-hashrate-escrow.md