The Coin Unchained Network

Protocol Version 0.1.0

Carol Schulze

May 30, 2021

Abstract

Coin Unchained is a novel concensus network based on the idea of *individual convergent concensus*, where nodes converge on a concensus without explicit agreement. Instead, every node has acces to the same shared, signed, append-only database with which all legitimate nodes use to arrive to compatible *local* conclusions, thereby converging on a concensus without the need for complex concensus protocols. Actors "pay" for participation in the network by verifying transactions and hunting for malicious actors, thus creating an environment highly resistant to rule-breakers while also avoiding useless work common to proof-of-work blockchains and complex proof-of-stake schemes.

1 Overview

Coin Unchained is organized in three levels:

- The overlay network: virtual network created by connected nodes. Unlike most P2P overlay networks, the topology is randomized and unremarkable due to gossip protocols.
- The database: the shared record of all *relevant* transactions in the network. Actors may be blacklisted for trying to subvert the rules, and in that case their records need not be kept.
- Ancillary Coins (A-Coins): the network currency. Unlike bitcoin or ether, A-coins cannot be traded and have a maximum cap for each actor. A-coins are required to use the network, and are gained by maintaining it.

Coin Unchained's network does not rely on traditional miners and their usefulness is limited. Instead, every user of the network must also contribute to it. Thus the health and integrity of the network is actively maintained by relevant participants, creating *confidence* in a trustless system.

Even in the face of sybil attacks, every transaction is verifiable thus making it feasible to weed out bad actors regardless of how much of the network they control. On the other hand, with increased confidence (and assuming *some* legitimate nodes in the network) it is possible for less strict applications to accept transactions as valid without actually verifying it as it will almost certainly be flagged as invalid in due time.

2 The Database

The whole network relies on a shared database. Every item in the database is a signed transaction that refers both to its author and an application (see section 6), plus extra metadata (see subsection 2.1). Transactions are ordered in a strict linear sequence, and any attempts to meddle with this sequence will lead to a ban from the network when detected (see subsection 5.1).

The design of this database is heavily based on Secure-Scuttlebutt[1]. We define the database as a collection of transactions. Each transaction contains, among other things, the ID of the actor that created it, the application for which the transaction refers, a monotonic sequence number, and a cryptographic signature the encompass the current and the previous transaction in the sequence.

Transactions form a linear sequence when grouped by (author ID, application ID). New

transactions are propagated through a simple gossip protocol where nodes push all transactions that other nodes are missing. To shut off write access to bad actors, each actor has their own list of (author ID, application ID) that it accepts, and all legitimate nodes should converge to the same list.

When detected, bad actors are blacklisted and their transactions purged. New actors also require a proof of work to reduce the impact of spamming.

Both sybil attacks and spamming are severely restricted. Once a transaction is created and spread to the rest of the network it is effectively commited. While it is possible (but extremely unlikely, see subsection 7.1) to control the flow of information throughout the network, once a transaction is received and verified by a node it is permanent. Short of breaking into a node and modifying its database, or breaking the underlying crypto, no amount of computing power, monetary power, or time allows an attacker to manipulate history or the validity of individual transactions. Furthemore, nodes will keep trying to spread the transactions they know about in a random fashion (see section 3), so confining transactions to any subsection of the network requires an effective and permanent split.

2.1 Transactions

A transaction is an authenticated object containing the following pieces of information:

- The ID of the transaction's author, as a SHA3-256 hash of their public key.
- The ID of the application as a transaction reference.
- A monotonic sequence number that must be increased by one for every new transaction in a (author ID, application ID) pair.
- A proof-of-work to reduce spamming.
- A message for the application.
- A cryptographic signature of the whole transaction.

2.2 Actors and Nodes

Trusted actors "introduce" new actors in their own transaction logs.

3 Overlay Network

4 Claims and Certification

Positive certification fulfills multiple purposes:

- It allows nodes to probabilistically accept the transaction as valid without needing to actually validate the transaction by themselves. This is useful for applications that can recover from invalid transactions that are only uncovered at a later date.
- Makes sure every transaction is validated by multiple actors, reducing the chance of invalid transactions. Invalid transactions can be certified by malicious actors, though having enough malicious nodes with enough trust to be able to certify an invalid transaction is expensive. It is also an extremely dangerous gambit as such attempts are almost guaranteed to be quickly uncovered and harshly punished by legitimate nodes in the network.
- Makes sure the transaction is actually known by multiple nodes, and manipulation attempts easier to detect and punish.

Plus, verification in general is necessary for a healthy network.

5 Emergent Properties

5.1 Trust

5.2 Vets

5.3 State

6 Applications

7 Attack Scenarios

7.1 Sybil Attacks

7.2 Timestamp Spoofing

8 Wants

Verifiable timestamp on transactions and transaction timeout.

References

[1] Anne-Marie Kermarrec, Erick Lavoie, and Christian Tschudin. Gossiping with append-only logs in secure-scuttlebutt. In *Proceedings of the 1st International Workshop on Distributed Infrastructure for Common Good*, DICG'20, page 19–24, New York, NY, USA, 2020. Association for Computing Machinery.