ChainCash - elastic peer-to-peer money creation via trust and blockchain assets

kushti, scalahub

November 12, 2023

Abstract

In this paper we introduce ChainCash, a protocol to create money in self-sovereign way via trust or collateral. The protocol allows for elastic money creation in peer-to-peer environment, without any authority, being it authoritarian, democractic or algorithmic. For that, acceptance of notes created by other peers is an individual choice.

1 Introduction

Currently, most of monetary value is created by private banks (often, offshore banks as in so-called "eurodollar" system [1]) following central banks requirements. As an alternative, starting with Bitcoin [2] launch in 2009, a lot of cryptocurrencies and DeFi applications on top of public blockchains are experimenting with algorithmic money issuance. As another option, we also have alternative, usually local, monetary systems, such as LETS (local exchange trading systems [3]), timebanks, local government currencies [4], and so on. Control in traditional fiat monetary systems is possessed by big players (with rich getting richer effect) creating money in non-transparent ways (especially in offshore circuits) without reasonable limits, on the other side, fiat monetary systems (in opposite to commodity money used before fiat, as well as alternative monetary systems) have best supply elasticity. Cryptocurrencies (and, sometimes, other tokens on top of public blockchains) have strict algorithms emitting new money, thus they have publicly known emission schedule, which makes them perfect digital commodity assets, on the other hand, supply is disconnected from economic activity and not elastic. Local currencies usually considered more fair in segniorage distribution than fiat currencies, they are successfully boosting local economies often, but, in opposite to fiat and crypto-currencies, they are not global and usually are dying without active core. Elasticity of supply is also limited simply due to entry barriers for external actors.

In this paper, we propose ChainCash, a new global kind of money, with decentralized issuance, elastic supply. ChainCash notes are collectively backed by collateral and trust. ChainCash acts on top of Ergo blockchain (possibly, other public blockchains as well, however, for efficient implementation there is requirement for primitives which can be found in Ergo only at the moment, to the best of our knowledge). Thus collateral for a ChainCash comes from reserves network peers may have, and on spending a note, a peer is attaching its reserve to collective backing. At the same time, a newly issued note could be accepted by a peer without any backing provided, for example, if such a note is issued by a friend or a trusted charity. Every peer in the system is having own individual rules for accepting notes (widely accepted standards may exist at the same time), which provides basis for elasticity of supply. We are providing details in the next section.

2 ChainCash Design

We consider money here via its medium-of-exchange property. For existing currencies, there are usually many options to represent value, such as coins, paper or plastic banknotes, digital records in different ledgers, etc. For ChainCash, we define money as a set of digital notes, each has some nominal (not fixed but arbitrary in our case). Value of a note is nominated in some existing widely recognizeable unit-of-account, for example, in milligrams of gold.

We consider that an economy is consisting of known agents $a_1, ..., a_n$. Then we can define medium-of-exchange property of money via a set of agents accepting monetary objects (i.e. notes). Usually, set of agents accepting some kind of money (e.g. local or foreign currency) is fixed. That is, for every monetary object (e.g. a note) which belongs to a fixed sort of money, an agent is accepting it as a mean of incoming payment, or reject. In opposite, for ChainCash money, similarly to [5], the set is individual for a note, so when agent a_i sees a note n, it applies its personal predicate $P_i(n)$ to decide whether to accept the note or decline it.

Then how notes are different in case of ChainCash? We consider that every note is collectively backed by all the previous spenders of the note. Every agent may create reserves to be used as collateral. When an agent spends note, whether received previously from another agent or just created by the agent itself, it is attaching its signature to it. A note could be redemeed at any time against any of reserves of agents previously signed the note. However,any agent after the first one in signatures chain is getting redemption receipt which is indicating debt of previous signers before him, and then he may redeem the receipt against a reserve of any previous signer, with a new redeemable receipt being generated. Also, redemption fee should be paid, and the fee is incentivizing reserves provision and also using the notes instead of redeeming them. The protocol does not impose collateralization requirements, it is allowed for an agent to issue and spend notes with empty reserve even. It is up to agent's counter-parties then whether to accept and so back an issued note with collateral or agent's trust or not.

As an example, consider a small gold mining cooperative in Ghana issuing a note backed by (tokenized) gold. The note is then accepted by the national government as mean of tax payment. Then the government is using the note (which is now backed by gold and also trust in Ghana government, so, e.g. convertible to Ghanaian Cedi as well) to buy oil from a Saudi oil company. Then the oil company, having its own oil reserve also, is using the note to buy equipment from China. Now a Chinese company has a note which is backed by gold, oil, and Cedis. It could be hard maybe for Chinese company to redeem from a small cooperative in Ghana, so it can redeem from Ghana government, and the government may redeem from the cooperative.

Agent's note quality estimation predicate $P_i(n)$ is considering collaterals and trust of previous spenders. Different agents may have different collateralization estimation algorithm (by analyzing history of the single note n, or e.g. all the notes issued by previous signers of n, other options are also possible), different whitelists, blacklists, or trust scores assigned to previous spenders of the note n etc. So in general case payment sender first need to consult with the receiver on whether the payment (consisting of one or multiple notes) can be accepted. However, in the real world likely there will be standard predicates, thus payment receiver (e.g. an online shop) may publish its predicate (or just predicate id) online, and then a payment can be done without prior interaction.

We propose to implement ChainCash monetary system on top of a public blockchain as:

- blockchain provides an instant solution for public-key infrastructure
- public blockchain allows for a global ledger solution with minimal trust assumptions [6]
- as a consequence, global public ledger allows for simple analysis of notes in existence
- smart contracts minimize trust issues in payment execution and redemption. If native blockchain currency and assets on top of it (such as algorithmic stablecoins) used in reserves, trust issues in redemption could be eliminated at all. If tokenized real-world commodities and fiat currencies (e.g. USDT) are used in reserves, redemption could not be completely trustless (as smart contracts do not have power off the chain), but at least there is transparent accounting in on-chain part of redemption

We use Ergo as a Proof-of-Work blockchain to implement ChainCash, as it is built on minimal trust assumptions [6], and UTXO transactional model as well as AVL+ trees support are making notes implementation feasible.

3 ChainCash Implementation

For blockchain-based ChainCash implementation, we consider how to implement two main parts:

- contracts for notes, reserves, and redemption receipts. Here, we consider on-chain contracts as the most straightforward option. Then we may consider more scalable option, such as having reserves (and maybe receipts) only on chain, and have notes making progress on a side-chain or off-chain (on top of some Layer 2 solution)

- client software (which we refer to as ChainCash Server as well), which is interacting with the blockchain (possibly, also a sidechain, or p2p network where notes are making progress off-chain). This software is implementing a_i agent's functionality from the Section 2, including note quality estimation predicate $P_i(n)$. For that, the client may potentially track all the reserves and notes. Client's $P_i(n)$ may be configured via whitelists, blacklists, collateralization requirements provided in config. ChainCash Server can be seen as a bank as in b̈e your own banküsed in Bitcoin community, however, in Bitcoin a node is a passive and indistinguishable from others bank just validating common history, while ChainCash Server has individual behavior defined by its config.

On-chain contracts are available at [7]. Three contracts can be found there, namely, reserve, note and receipt contracts. Reserve contract locks ERG native tokens on top of Ergo blockchain and allow to redeem native or custom tokens when a note is presented. Note contract ensures that the note has proper history, that is, on every spending a valid signature of corresponding reserve is added. It also and allows for a note to be split into two parts (payment and change), and allows for note redemption.

Reference ChainCash Server implementation (in Rust programming language) can be found at [8].

Basic contracts implementation described is good for starters, but can be extended in many ways. We note that it is possible to add new features without need for the whole network to update. New features, such as new reserve and note contracts, can be proposed in form of CCIPs (ChainCash Improvement Proposals). ChainCash Server may support new features, in particular, new forms of notes. If client is asked to accept a note with unknown contract, or a note backed by unknown contract, it is just refusing to accept the note.

4 Applications

ChainCash could be seen as a powerful foundation for other monetary systems, and we are going to show it in this section.

4.1 LETS

To implement a local exchange trading system on top of ChainCash, every LETS member needs to whitelist every over member, so they will accept notes of each other regardless reserves backing the notes, and thus LETS can create money within the community (the LETS circle) with no limits. On the other hand, unlike traditional LETS, notes can circulate outside the LETS circle easily as well. Implementations may vary from LETS members whitelisting unconditionally only notes issued by other members to members whitelisting notes ever signed by LETS members.

4.2 Local Currencies

A local or even national government may issue notes and enforce their acceptance within its jurisdiction by enforcing economic agents to accept notes issued or spend by the government. As well as in a LETS implementation, enforced acceptance rules may vary.

Often local currencies are introducing redemption fee, to promote local usage. In ChainCash, similar goals can be achieved via modifying the reserve contract in a way that non-locals need to pay redemption fee while locals need not, alternatively, the note contract could be modified in a way that spending to non-local addresses incurs a fee. Local currencies are often associated with demurrage, after well-known Woergl experiment [4]. Demmurage could be implemented by modifying note contract. However, modifying note contract makes notes locked by it less appealing to others, but that is common for local currencies already.

4.3 Multilateral Trade-Credit Set-off

Multilateral Trade-Credit Set-off **??** is a technique which allows invoices in closed loops to be cleared against one another. In ChainCash, it is possible to clear mutual debts by just burning atomically tokens backed by counter-partis in a single transaction. This will allow them to issue more notes after.

5 ChainCash Advantages and Drawbacks

In this section, we are providing some thoughts on possible advantages and drawbacks of ChainCash. Note that practice can show completely different picture from what we are providing here (as often happens).

Advantages:

-
-

Drawbacks:

- ChainCash notes are non-fungible, while they share the same unit-of-account, each note has unique backing. This prevents ChainCash usage in many DeFi applications, such as liquidity pools, lending pools etc. We note that, similarly, DAI stablecoins issued against CDPs (collateralized debt posistions) with different collateralization also should be priced differently. And like DAI is assigning the same price to DAIs of different quality, there could be services on top of ChainCash combining notes of certain quality, buy e.g. exchanging them with service tokens which then can be used in DeFi services.

- There is no privacy in ChainCash payments now. This topic is fully left for further research.

References

[1] F. Machlup, "Euro-dollar creation: A mystery story," *PSL Quarterly Review*, vol. 23, no. 94, 1970.

[2] S. Nakamoto and A. Bitcoin, "A peer-to-peer electronic cash system," *Bitcoin.–URL: https://bitcoin.org/bitcoin.pdf*, vol. 4, no. 2, 2008.

[3] C. C. Williams, "The new barter economy: an appraisal of local exchange and trading systems (lets)," *Journal of Public Policy*, vol. 16, no. 1, pp. 85–101, 1996.

[4] M. Unterguggenbercer, "The end results of the woergl experiment.," *Annals of Public and Cooperative Economics*, vol. 10, no. 1, pp. 60–63, 1934.

[5] K. Saito, "Peer-to-peer money: Free currency over the internet," in *Web and Communication Technologies and Internet-Related Social Issues—HSI 2003: Second International Conference on Human. Society@ Internet Seoul, Korea, June 18–20, 2003 Proceedings 2*, pp. 404–414, Springer, 2003.

[6] "Know your assumptions." `https://www.ergoforum.org/t/know-your-assumptions/4198`. Accessed: 2023-01-30.

[7] "Chaincash contracts." `https://github.com/ChainCashLabs/chaincash/tree/master/contracts/onchain`. Accessed: 2023-11-11.

[8] "Chaincash server." `https://github.com/ChainCashLabs/chaincash-rs`. Accessed: 2023-11-11.