Sovereign Personal Cryptocurrencies:
A Grassroots Foundation for a Digital Economy

Ehud Shapiro
Weizmann Institute of Science, Rehovot, Israel
ehud.shapiro@weizmann.ac.il

Abstract. Sovereign cryptocurrencies are a digital means for turning trust into liquidity. They are units of debt that can be issued and traded digitally by people, communities, corporations, municipalities, and central banks, and can serve as a foundation for a grassroots digital economy that emerges and functions without initial capital or external credit. As each sovereign provides for the economic and computational integrity of its own cryptocurrency, the distributed implementation of sovereign cryptocurrencies needs only realize dissemination and leader-based equivocation (double-spending) exclusion. Importantly, it does not require expensive all-to-all synchronization protocols such as blockchain consensus or Byzantine Atomic Broadcast, nor even Byzantine Reliable Broadcast. In this paper we introduce the principles that underlie sovereign cryptocurrencies, and focus on sovereign personal cryptocurrencies, issued and traded by people: We elaborate their possible economic uses, as derived from these principles; specify them formally via multiagent transition systems; prove the resulting protocol to be grassroots, meaning that disjoint instances of it can be deployed independently and later interoperate; specify a grassroots distributed implementation of the sovereign personal cryptocurrencies protocol; and prove it correct. The difference between sovereign cryptocurrencies for people and for organizations is that people can issue and trade sovereign coins on their own behalf, whereas signatures by multiple people are normally required to authorize a transaction on behalf of an organization. Elaborating the principles of sovereign cryptocurrencies for organizations and extending the specification and implementation to multisignature accounts is the subject of subsequent work.

1 Introduction

Concept. Sovereign cryptocurrencies are units of debt that can be issued and traded digitally by people as well as by organizations such as partnerships, corporations, cooperatives, community banks, municipalities, and central banks. They are based on four principles:
1. Sovereignty: A sovereign can issue and transfer coins at its discretion.
2. Pricing: A sovereign endows its coins with value by pricing its offerings—goods and services for people and corporations; taxes for cities and states; and fiat currencies for central banks—in terms of its sovereign coin.
3. **Mutual Credit:** Mutual credit lines with ensuing liquidity are formed by the voluntary exchange of sovereign coins among sovereigns with mutual trust: Among family, friends and peers; between a community bank and its members; between a corporation and its employees, suppliers and customers; between federal, state and municipal governments; and between a central bank and local banks, businesses and residents.

4. **Coin Redemption:** A sovereign is obligated to redeem any coin it has issued—upon a claim by its holder—against any sovereign coin it holds. Coin redemption can: (i) Make coins by the same sovereign fungible; (ii) resolve double-spending; (iii) let a credit provider revoke or shift credit; (iv) allow chain payments across mutually-liquid sovereign cryptocurrencies; and (v) make the exchange rate among all mutually-liquid sovereign cryptocurrencies one-to-one.

Sovereign cryptocurrencies forsake external capital or credit, allowing communities to leverage mutual trust into a liquid grassroots digital economy; it allows corporations and cooperatives to form without initial capital by establishing mutual credit lines with their employees, trusted customers and trusting suppliers; it affords all levels of government—if liquid—to endow liquidity to their local economies and tame it; and allows mutually-liquid sovereigns to trade internationally. **Insolvency** is the inability of a sovereign to fulfill redemption claims against it due to lack of liquidity, manifest by not holding coins of other sovereigns, in particular coins of its creditors. An insolvent sovereign can regain liquidity by selling enough goods and services and/or receiving additional credit (coins) from others to settle all outstanding redemption claims against it.

The responsibility for the economic and computational integrity of a sovereign cryptocurrency resides with the sovereign—person or organization—that issued it. As the value of a sovereign currency depends on such integrity, there is no reason to separate the responsibility for the economic integrity and the computational integrity of a sovereign cryptocurrency: Both reside with the sovereign.

The Byzantine fault against which cryptocurrencies seek defense is equivocation, or double-spending. A sovereign has no incentive to double-spend its coins, nor to ratify the double spending of its coins by others; in both cases the sovereign would be better off issuing new coins and handing them over instead. Thus, sovereign coins can have a simple grassroots implementation that need only realize local dissemination and leader-based equivocation (double-spending) exclusion, where each sovereign provides transaction finality to their currency holders. Importantly, the implementation does not require expensive all-to-all synchronization protocols such as blockchain consensus or Byzantine Atomic Broadcast, nor the simpler Byzantine Reliable Broadcast.

In this paper we focus on sovereign personal cryptocurrencies. The difference between those and sovereign organizational cryptocurrencies is technical, not fundamental: People can issue and transfer sovereign coins on their own behalf, whereas signatures by multiple people are normally required to authorize transactions on behalf of an organization. Elaborating the principles of sovereign cryptocurrencies for organizations and extending the specification and imple-
mentation of sovereign cryptocurrencies to multisignature accounts is the subject of subsequent work.

**Background and related work.** The non-fungible tokens (NFTs) trade protocol presented here is an abstract counterpart of asset-transfer objects [13]. While both mainstream and community cryptocurrencies require total ordering for their realization and hence employ Nakamoto consensus/Byzantine Atomic Broadcast, NFT trade can make do with the strictly-weaker equivocation exclusion [14]. Sovereign cryptocurrencies are at the bottom of an economic hierarchy corresponding to the established hierarchy [14] in distributed computing: (i) **Dissemination** (e.g., [9]) with leader-based equivocation exclusion [31] is sufficient for implementing sovereign cryptocurrencies, as their longevity need not exceed that of the sovereign. (ii) **Reliable broadcast** [4], supermajority-based equivocation exclusion [13,33,7,3] is necessary for implementing an NFT trade protocol, as in general we wish the ownership of an NFT to outlive its creator, especially for art NFTs. (iii) **Ordering**, based on blockchain consensus, State-Machine Replication, or Byzantine Atomic Broadcast [5,40,17], is used for mainstream cryptocurrencies. Importantly, these protocols are not grassroots [31], unlike the implementation of sovereign cryptocurrencies presented here, which is.

Mainstream cryptocurrencies—based on proof of work [22] or stake [19]—grant participants power and wealth in accordance with their capital investment, thus exacerbate economic inequality. Community cryptocurrencies attempt to redress this: They aim to achieve social goals such as distributive justice and Universal Basic Income (UBI) [23,16,2], for example by egalitarian coin minting [29,24]. But, like mainstream cryptocurrencies, they presently rely on execution by third parties—miners that require remuneration for their service. Hence, neither mainstream cryptocurrencies nor extant community cryptocurrencies can fulfill a key requirement of a grassroots digital economy—bootstrap without external capital or credit. Sovereign coins are a special case, or an instance, of an abstract notion of NFTs. NFTs emerged in the context of the Ethereum blockchain [39], to contrast them with the coins of the cryptocurrency that are fungible.

The idea of a personal cryptocurrency was floated almost a decade ago [26]. Circles [6] developed a UBI-based personal currency system, but it is not sovereign in several respects: It has an enforced uniform minting rate, an enforced 1:1-exchange rate among currencies of people who trust each other, and execution by third-party miners on an Ethereum platform. More recently, a platform for ‘celebrity coins’ named Promify was launched [25]. It enabled artists/celebrities (defined as having at least 10,000 followers on social media) to issue their personal coins, and sell their fans exclusive or preferred access to content or merchandise owned by the artist in exchange for said coins. The Promify platform provides the economic framework and computational support in exchange for transaction fees and appreciation of the underlying coin [25]. We share the basic concept that a key economic function of a personal currency is to purchase goods and services from the person issuing the currency, and Promify provides a proof of the viability of this concept. Yet, celebrity coins controlled by a third party are
not a suitable starting point for the design of sovereign personal cryptocurrencies for a grassroots economy.

Economically, sovereign cryptocurrencies are closer to national fiat currencies than to mainstream cryptocurrencies, in that their sovereign—the state, corporation, community, or person—has complete control on minting new coins. Thus, the following aphorism holds: ‘everyone can create money; the problem is to get it accepted’ [21]. Thus, the dynamics of a sovereign cryptocurrency is dictated first and foremost by its sovereign, with basic concepts of international monetary economics [34] such as foreign debt, trade balance, and currency velocity being directly relevant to their analysis, as discussed below (#7).

In a sense, a personal cryptocurrency is complementary to the original Ripple idea [11], of digital money as IOUs. Here, a sovereign coin is a unit of debt that can be used either to purchase goods or services offered by the sovereign, or to swap it with another unit of debt owned by the sovereign. The effect of redemption claims is similar to coin equality among trusting members of Circles UBI [6,10], but different in that it is not UBI based. Establishing mutual credit via personal cryptocurrency exchanges results in mutual credit lines as in Trustlines [15] and credit networks [8,12], allowing the body of research on liquidity in credit networks to carry over to personal cryptocurrency networks.

The term ‘self-sovereign digital identities’ [1] was proposed with the idea that the person, rather than an organization or the government, should be the sovereign of their digital identity [38]. It is being developed into a set of standards by the W3C [37]. Sovereign personal cryptocurrencies extend this vision to the realm of cryptocurrencies; following this terminology, they might also be called ‘self-sovereign cryptocurrencies’.

Outline. Section 2 elaborates the possible economic behaviors of sovereign cryptocurrencies when issued and traded by people, as derived from their principles. Section 3 specifies sovereign personal cryptocurrencies as a special case (formally, a subset) of an NFT trade protocol, which is specified in turn via multiagent transition systems (Appendix 7). It introduces the notion of coin redemption, explores synchronous and asynchronous interpretations to the desideratum that redemption claims be responded to promptly, defines sovereign personal cryptocurrencies based on the asynchronous interpretation, and proves the protocol to be grassroots. Section 4 presents an implementation of sovereign personal cryptocurrencies in terms of an asynchronous block dissemination protocol and proves the implementation correct. Section 5 concludes. Proof are relegated to Appendix 6. The presentation employs the mathematical framework of multiagent transition systems [31]; their definitions and results used here are replicated in Appendix 7.

2 Principles and Scenarios

Sovereign cryptocurrencies are a conceptual and mathematical construction. To explore its implications, we elaborate on hypothetical scenarios for the use of sovereign cryptocurrencies by people, based on their mathematical properties.
Here we elaborate the principles of sovereign cryptocurrencies as applied to personal cryptocurrencies, stated in terms of a sovereign person $p$.

1. **Personal Coin Transactions**: The person $p$ may issue new $p$-coins and may transfer any coin it holds to other people at its discretion. A sovereign coin includes a record of its provenance, which is extended with every transfer. A double-spending occurs when a person transfers the same coin to two different people, resulting in the coin having two inconsistent provenances with a shared prefix. A sovereign has no incentive to double-spend their own coins and undermine their reputation, as they can issue additional coins instead.

2. **Personal Coin Pricing**: People endow their coins with value by pricing their goods and services in terms of their personal coins. Examples may include ‘babysitting for you this Friday evening’ for 5 $p$-coins, ‘the use of my apartment this weekend’ for 120 $p$-coins, ‘a lift to the beach this morning’ for 3 $p$-coins, ‘use of my car tomorrow’ for 80 $p$-coins, or ‘ownership of my car’ for 20,000 $p$-coins.
   In addition, $p$ may price $p$-coins, e.g. in terms of fiat coins, in case there are no $p$-coins in circulation for a person that wishes to purchase good or services from $p$. The net effect of two such transactions (purchase $p$-coins from $p$ for fiat coins, and then use the $p$-coins to purchase goods or services from $p$) is a purchase with fiat coins; in a liquid economy without ‘cash hoggers’ (i.e. people who consistently produce and sell more than they consume) such transactions would be the exception.
   We note that trading sovereign coins for ‘off-chain’ goods or services is not atomic: It requires trust and entails a risk, bore by the payer and/or the payee of the sovereign coins, depending on whether the payment occurs before or after transferring the goods or performing the service. Such a risk can be mitigated using a multisignature escrow account shared with a trusted third party.

3. **Personal Coin Exchange**: Mutual credit lines with ensuing liquidity are formed by the voluntary exchange of personal coins among family members, friends, peers, and colleagues. For example, in a local community of 501 people, if every two people exchange 100 coins with each other, each person will have 50,000 coins of other community members, while issuing and transferring 50,000 of its own coins to others, achieving liquidity for the entire community without external capital or credit. Such liquidity may boost the community’s local economy, with the exposure of every community member to every other member initially limited to 0.2% of their total liquidity.
   Note that in general a mutual line of credit requires mutual agreement, the negotiation of which may produce an asymmetric result: A high-liquidity person $p$ may charge a premium for its credit to a credit-less newcomer $q$, by providing less than one $p$-coin for each $q$-coin received. Such a premium may turn into a profit for $p$ once the newcomer $q$ establishes independent liquidity. See discussion of creditworthiness (#7).
4. **Personal Coin Redemption:** A person is obligated to redeem any coin it has issued—upon a claim by the person that holds the coin—against any personal coin it holds, whether issued by self or by others. Coins redemption has many ramifications:

(a) **Coin redemption sub-cases:** When \( p \) redeems a \( q \)-coin from \( q \) it may request:
   i. A \( p \)-coin \( q \) holds; to revoke credit given to \( q \).
   ii. A fresh \( q \)-coin; to reset provenance and thus reduce the risk of a coin having been double-spent, or to resolve its double-spending.
   iii. An \( r \)-coin \( q \) holds, \( p, q \neq r \); for chain payments, arbitrage, risk management.

All these uses are discussed below.

(b) **Revoking credit:** A mutual credit line between \( p \) and \( q \) is established voluntarily (\#3 above), but can be revoked unilaterally, by \( p \) redeeming a \( q \)-coin it holds against a \( p \)-coin \( q \) holds, if \( q \) has any left; and vice versa. Note that if a mutual credit line between \( p \) and \( q \) is established with a premium for \( p \) and then immediately revoked by \( p \) via one-to-one coin redemptions, then \( p \) remains with the premium \( q \)-coins and \( q \) is left with nothing. This is analogous to a loan with upfront interest payment that is recalled in full immediately after interest has been paid. To avoid this, an agreement on an asymmetric mutual credit line should also include a limitation on when credit can be revoked. Such an agreement can be enforced with a multisignature escrow account.

(c) **Make all coins issued by a person fungible:** A holder \( q \) of a \( p \)-coin \( c \) may redeem \( c \) from \( p \) for a fresh \( p \)-coin \( c' \). This could be beneficial in case the coin \( c \) has a long provenance and/or its provenance has people that \( q \) does not know or trust. Hence, if \( q \) receives \( c \) as part of a transaction, then before completing the transaction \( q \) may redeem \( c \) for a fresh \( p \)-coin \( c' \), with a provenance stating that \( c' \) was created by \( p \) and then transferred to \( q \). Since \( p \) has no reason to double-spend its coins, then \( q \) can now complete the transaction, without a concern regarding the provenance of the \( p \)-coin it received.

(d) **Resolve double-spending:** Redeeming a \( p \)-coin against a fresh \( p \)-coin also resolves double-spending, as the sovereign \( p \) would redeem a \( p \)-coin at most once. If, in the example above, the \( p \)-coin \( c \) was double-spent, and another person has redeemed \( c \) from \( p \) before \( q \), then \( p \) would not consider \( q \) an owner of \( c \) and refuse \( q \) to redeem \( c \), much the same way that a second transaction of a double-spend is ignored in standard blockchain protocols. Any \( p \)-coin with a provenance longer than one can be redeemed from \( p \) to exclude double spending; however, if the payee sufficiently trusts the people on the provenance chain, such an extra step of communication-and-synchronization with the sovereign can be spared. Needless to say, even if double spending is not (yet) outlawed, the reputation of a double-spender would be tarnished with grave consequences.
for their trustworthiness, creditworthiness (see #7), and hence for the value of their sovereign currency.

(e) **Entail price-equality of coins issued by mutually-liquid people:**

If \( p \) and \( q \) hold each other’s coins, then \( p \) can redeem any \( q \)-coin it holds against a fresh \( q \)-coin, as well as against any other person’s coin held by \( q \), and vice versa. Thus, coin redemption among mutually-liquid people allows arbitrage and thus entails equality of the price of their coins.

(f) **Enable chain redemptions:** Assume that there is **1-liquidity from \( p \) to \( q \)**, defined by the existence of a sequence of people \( p_0, \ldots, p_k \), \( p_0 = p \), \( p_k = q \), where each \( p_i \), \( 0 \leq i < k \) holds a \( p_{i+1} \)-coin \( c_{i+1} \). Then \( p \) can initiate chain of \( k - 1 \) redemptions, where the \( i^{th} \) redemption claim by \( p \) to \( p_i \), \( 0 < i < k \), transfers the coin \( c_i \) in return for the \( p_{i+1} \)-coin \( c_{i+1} \).

(g) **Enable long-range arbitrage and chain payments:** Chain redemption can be used for long-range arbitrage and chain payments. If the price of a \( q \)-coin is higher than the price of a coin \( c \) held by \( p \), and there is 1-liquidity from \( p \) to \( q \) that starts with \( c \), then \( p \) can exchange \( c \) for a \( q \)-coin via chain-redemption, and rake in the difference in their price. If \( p \) wants to purchase something from \( q \) in return for a \( q \)-coin but does not hold any, and there is 1-liquidity from \( p \) to \( q \), then \( p \) can trade a coin it holds for a \( q \)-coin via chain-redemption, and then use the \( q \)-coin it now holds to pay \( q \).

(h) **Allow risk management:** In the mutual-credit community described above (#3), if a person \( p \) has a reason to believe that \( q \) is not creditworthy, then as long as \( q \) is still liquid, \( p \) may reduce its exposure to \( q \), while retaining its overall liquidity, by redeeming a \( q \)-coin it holds against an \( r \)-coin held by \( q \), for some person \( r \) that \( p \) considers more creditworthy than \( q \).

5. **Mutual Credit Lines are a Sybil-Repellent:** Preventing sybils (fake or duplicate digital personas) from joining a digital community is a major open challenge [27,53,86]. One approach [28,23] employs mutual sureties in sybil-resilient algorithmic admission of people to a digital community. Such mutual sureties should be backed by some cryptocurrency. But, if a grassroots digital community depends on cryptocurrency-backed mutual sureties to grow, how can it grow without initial capital or credit? We saw above (#3) that mutual sureties can provide a growing community with liquidity without initial capital or credit. It so happens that mutual credit lines are precisely the mutual sureties needed for sybil-resilience. Mutual credit lines are established between people who know and trust each other, so the risk of a fake identity is low. Regarding duplicate identities, a person \( p \) obtaining credit from different people using different digital identities, say \( p' \) and \( p'' \), in effect hides assets from its creditors—family, friends, and colleagues—since holders of a \( p' \)-coin cannot redeem it against assets held by the \( p'' \) account, and vice versa, as long as it is not publicly known that \( p' \) and \( p'' \) identify the same person \( p \). Obtaining credit from people close to you under a false pretence and while hiding assets from them is a major breach of the trust
of these people, an act that cannot be perpetrated by most. And the few that do will be hard-pressed to maintain two different digital identities, each initially known to only half of one’s social circle, without being exposed. Hence, mutual sureties may be an excellent tool for repelling sybils from digital communities.

This capability of personal cryptocurrencies provides a major justification for using them as the basis for a grassroots digital economy, as opposed to commencing with community/bank cryptocurrencies: A community bank, when admitting a new member, will evaluate their credit standing with the member’s peers: Employing mutual lines of credit among community members as collateral for the credit line provided by the bank is precisely what would make the community bank sybil-resilient.

6. **Liquidity**: Credit networks have been studies extensively [8,12], and have inspired the initial design of some cryptocurrencies [11]. The key instrument of credit networks is chain payments. The fundamental difference between credit networks and sovereign cryptocurrencies is manifest when liquidity is constrained: Credit networks assume an outside measure of value (e.g. a fiat currency) in which credit units are given, whereas sovereign cryptocurrencies, being sovereign, are not linked *a priori* or during their lifetime to any external unit of value. As a result, coins of a liquidity-constrained sovereign cryptocurrency devaluate naturally. However, as discussed above (#3), mutual lines of credit can be established via the exchange of sovereign coins, which together with coin redemptions provide for chain payments, just as in credit networks. Hence, much of the body of knowledge regarding liquidity in credit networks can be readily applied to sovereign cryptocurrencies, with the caveat that lack of liquidity not only causes chain payment failures, as in credit networks, but also devalues the liquidity-constrained sovereign cryptocurrencies involved.

7. **Creditworthiness**: Determining creditworthiness is an art. Still, personal cryptocurrencies offer objective measures for assessing the creditworthiness of a person $p$, analogous to international monetary economics [34]:

(a) The *foreign debt*, or more precisely and conversely, the *net interpersonal investment position* of $p$: The difference between the number of non-$p$ coins held by $p$ and $p$-coins in circulation (not held by $p$). Larger is better.

(b) The *trade balance* of $p$: The difference between the number of personal coins transferred to $p$ and the number of personal coins $p$ transferred (in any denomination) over a given period. Larger is better.

(c) The *velocity* of the currency of $p$: The number of $p$-coins transferred divided by the average number of $p$-coins in circulation, during a given period. Larger is considered better.

These three measures provide a rich foundation for objectively assessing the creditworthiness of a person, e.g., by a bank that assesses the risk of granting a credit line or a loan to a person, as well as by one person negotiating the premium in a mutual credit line with another person. This direct analogy
between personal cryptocurrencies and international monetary economy is a consequence of—and an attestation to—the sovereignty of personal cryptocurrencies.

8. **Private Banking:** A respected member \( p \) of the community, preferably with initial capital, may aim to operate as the community banker, following the footsteps of the Medicis and Rothschilds. By opening mutual lines of credit with all creditworthy members of the community, the banker’s personal currency can become the *de facto* community currency, as all transactions in the community can be completed by a two-step chain payment via the banker: If \( q \) wishes to pay \( r \)—both with mutual liquidity with the banker \( p \)—then \( q \) can redeem a \( p \)-coin it has in exchange for an \( r \)-coin the banker has, and then use it to pay \( r \). Naturally, the banker may choose to exert transaction fees and/or interest on drawn credit. Using multisignature accounts, a community may sidestep the Medicis and the Rothschilds by establishing a community bank, owned, operated and governed by the community, that issues a community currency.

9. **Insolvency:** A person \( p \) may reach insolvency—not holding any non-\( p \)-coins—by spending all the credit it has received without generating sufficient income in tandem. The dynamics of insolvency offers a ‘built-in’ natural path for recovering from it: The longer a person \( p \) is insolvent, its creditors, namely the holders of \( p \)-coins, will have less faith in—or patience for—\( p \), and hence may offer to sell \( p \)-coins at an ever-growing discount. The lower the price of \( p \)-coins, the easier it would be for \( p \) to recover from its insolvency. Since \( p \) prices its goods and services in \( p \)-coins, the lower the value of \( p \)-coins, the higher \( p \) can price its goods and services in \( p \)-coins and still be competitive. The higher the nominal price of \( p \)'s goods and services, the faster \( p \) can accumulate funds to settle the outstanding redemption claims against it and recover from insolvency, at which point the value of its personal coins returns to market value—the same value of coins of people it has mutual liquidity with. Note that those who would take a hit in such a scenario are the creditors of \( p \) who sold their \( p \)-coins at a discount.

10. **Bankruptcy and Multiple Identities:** An insolvent person may opt to escape its anguish by abandoning its digital identity and starting afresh, not unlike bankruptcy. However, bankruptcy is a regulated process that aims to mitigate the damage to creditors, whereas shedding one’s digital identity and assuming another is akin to fleeing the country in order to escape creditors. In both cases, the act leaves one’s creditors high and dry. To avert that, a mechanism to deal with duplicate identities (Sybils) of the same person is needed. While the notion of sybil identities and sybil-resilience is deep and wide [23,27,35,36], we note that an essential aspect of it, as it relates to sovereign personal cryptocurrencies, must be equating all identities of a person for the purpose of redemption: If the holder of \( p \)-coins discovers that \( p \) and \( p' \) are identities of the same person, then it may rightfully try to redeem its \( p \)-coins from \( p' \). Doing so would allow creditors to pursue the assets held by the indebted person under any of their duplicate identities.
11. **Death**: Any individual faces inevitable death, following which it cannot
consume or produce. The net interpersonal investment position of a person \( p \)
upon death determines whether the creditors of \( p \) can redeem their \( p \)-coins or
suffer a loss. We leave open the question of inheritance and the management
of the estate of a personal cryptocurrency account, noting that multisig
accounts are well-suited to implement any method devised to address it.

3 Specification

Here we specify sovereign personal cryptocurrencies as an instance of an abstract,
grassroots, Non-Fungible Tokens (NFT) trade protocol.

3.1 A Grassroots NFT Trade Protocol

For specifying sovereign personal cryptocurrencies, we consider single-signature
NFTs and identify people (that sign the creation and transfer of NFTs) with
their personal accounts (that receive and hold NFTs).

We assume a set of agents \( \Pi \), each equipped with a unique and single key
pair, identify an agent \( p \in \Pi \) with its public key, use \( P \) to denote an arbitrary
subset of the agents, \( P \subseteq \Pi \), and let \( S \) denote the set of all strings (sequences of
bits). Any agent can turn any string into an NFT that it holds, and can transfer
any NFT it holds to another agent, which becomes the new holder of the NFT.

We sidestep the question of whether an agent that creates an NFT infringes on
the rights of other agents in the NFT’s string.

**Definition 1 (NFT, Object, History, Provenance).** The structure \( x \xrightarrow{s} q \)
signed by \( p \), denoted by \( (x \xrightarrow{s} q)_p \), records a transfer of the payload \( x \) and
metadata \( s \) by the sender \( p \) to the recipient \( q \). The set \( T(P) \) of NFTs over
\( P \subseteq \Pi \) is defined inductively as follows:

1. **Object NFT**: The structure \( (x \xrightarrow{\text{initial}} p)_p \in T(P) \) for every string \( x \in S \)
   and agent \( p \in P \), shortened to object NFT, \( p \)-object, or object.

2. **Transfer NFT**: If \( x \in T(P) \) with recipient \( p \in P \) then \( (x \xrightarrow{s} q)_p \in T(P) \) for
every string \( s \neq \text{initial} \in S \) and every \( q \in P \).

Given an NFT \( x \in T(P) \), the history of \( x \) is the NFTs \( x_1, x_2, \ldots x_k \), \( k \geq 1 \),
where \( x_1 \in T(P) \) is an initial NFT, \( x_k = x \), and for each \( i \in [k-1] \), \( x_i \in T(P) \)
is the payload of \( x_{i+1} \); and if \( i > 1 \) then \( x_1 \) is the object of \( x \). The provenance
of \( x \) is the sequence of agents \( p_1, \ldots, p_k \), where \( p_i \in P \) is the recipient of \( x_i \).

For example, the following NFT ‘coin201’ \( p \xrightarrow{\text{initial}} \xrightarrow{\text{love}} p' \xrightarrow{\text{PO157}} p'' \) has a
\( p \)-object NFT with the string ‘coin201’, transferred by \( p \) to \( p' \) for love and then
transferred by \( p' \) to \( p'' \) to fulfil purchase order 157, with provenance \( p, p', p'' \).

The metadata \( s \) could be a request to obtain a product or service for which
the payment was made, a sales quote, or a purchase order. In a more complex
example, a contract \( c \) is signed by the parties \( p \) and \( q \) to exchange NFT \( x \) for NFT
\( y \), in which case \( s \) may be ‘transferred according to contract c’. For the transfer of
Definition 2 (NFT Consistency, Double-spending, Consistent and Complete Sets). Two sequences are consistent if one is a prefix of the other. Two NFTs with the same object are consistent if their provenances are consistent, else inconsistent. Two inconsistent NFTs constitute a double-spend by $q$ if their transfers NFTs at the first point of inconsistency are both signed by $q$. For any $P \subseteq \Pi$, a set $X \subset \mathcal{T}(P)$ of NFTs over $P$ is consistent if it is pairwise consistent; it is complete if for every NFT $x \in X$, $X$ includes the history of $x$.

In a consistent set of NFTs, the holder of each object is well defined:

Definition 3 (Object Holder). A consistent set of NFTs $X \subseteq \mathcal{T}(P)$, $P \subseteq \Pi$, defines a holding function that maps every object in $X$ to its holder in $P$, as follows: For every object $y \in X$, the holder of $y$ in $X$ is the recipient of the maximal-provenance NFT in $X$ with object $y$.

Note that the holder of an object in $X$ is well-defined since $X$ is consistent, even thought it may not be complete.

Sovereign personal cryptocurrencies are an instance of the following NFT trade protocol, the essence of which is that agents only transfer NFTs they hold, and only to agents they know, resulting in configurations that are complete and consistent, and in particular do not include double-spending and hence induce well-defined holdings. We adopt the notion of [31] of a protocol being a family of multiagent transition systems (see Appendix 7 for definitions). For $P \subseteq \Pi$, $\mathcal{T}^*(P)$ denotes the set of sequences over the set of NFTs $\mathcal{T}(P)$.

Definition 4 ($\mathcal{NT}$: NFT Trade Protocol). The NFT trade protocol $\mathcal{NT}$ over $\mathcal{T}^*$ has for each $P \subseteq \Pi$ a distributed transition system $\mathcal{NT} = (P, \mathcal{T}^*(P), c_0, TT, \lambda) \in \mathcal{NT}$ with configurations over agents $P$ and local states $\mathcal{T}^*(P)$, initial configuration $c_0 : = \{\Lambda\}^P$, and every $p$-transition $c \rightarrow c' \in TT_p$, $c' = c_p \cdot y$, for every NFT $y = (x \xrightarrow{q} q) \in \mathcal{T}(P)$, and every $p, q \in P$, where either:

1. $p$-Creates-$y$: $y$ is a $p$-object NFT, $p = q$, or
2. $p$-Transfers-$y$: $p$ is the holder of the object of $y$ in $c$.

The liveness condition $\lambda$ is the multiagent partition, $\lambda = \{TT_p : p \in P\}$.

Note that the object of $y$ above is $x$ if $x$ is an object; else, recursively, it is the object of $x$. Also, the liveness condition means, informally, that each agent individually is live. We overload an $\mathcal{NT}$ local state $c_p$ to also mean the set of NFTs in the sequence, and overload an $\mathcal{NT}$ configuration $c$ to also denote the union of these local sets.

Observation 1 $\mathcal{NT}$ is asynchronous.
Proposition 1 (Consistency and Completeness of NT Configurations).
Every configuration of every correct NT run is consistent and complete.

A grassroots protocol [31] can be deployed as independent and disjoint instances and over time, which can subsequently interoperate once interconnected. We note that client-server/cloud systems, blockchain protocols with hardcoded seed miners, e.g. Bitcoin [10], and permissioned consensus protocols with a predetermined set of participants, e.g. Byzantine Atomic Broadcast [32,18], are all not grassroots.

Theorem 1. NT is grassroots.

3.2 Sovereign Personal Cryptocurrencies as a subset of the NFT Trade Protocol

Sovereign cryptocurrencies, SC, are a ‘special case’ (formally, a subset [31]) of the NFT trade protocol NT. In particular, SC employs sovereign coins, which are NFTs with objects containing only strings of the form coin_i, i ∈ N.

Definition 5 (Coin Object and Transfer, Fresh, Sequence). A string of the form coin_i, i ∈ N is a coin string. A p-object NFT x that includes a coin string is referred to as a sovereign p-coin object NFT, p-coin object, or coin object for short. A transfer NFT is a p-coin transfer NFT, p-coin transfer, or coin transfer for short, if its object is a p-coin object, and is a fresh coin transfer if its payload is a coin object. A p-coin NFT, coin NFT, p-coin, or coin for short, is a p-coin object or a p-coin transfer. Let \( C(P) \) denote the set of all coins with provenance in \( P \subseteq \Pi \) and let \( C^\ast \) denote the function that maps a set of agents \( P \subseteq \Pi \) to the set of sequences of coins \( C(P)^\ast \).

Note that coin \( C(P) \) are the subset of NFTs \( T(P) \) with object strings being coin strings. For clarity, we repeat the definition of holder (Def. 3) for coins. In a consistent set of coins, the holder of each coin object is well defined:

Definition 6 (Coin Object Holder). A consistent set of coins \( X \subseteq C(P) \) defines a holding function that maps every coin object in \( X \) to its holder in \( P \), as follows: For every coin object \( y \in X \), the holder of \( y \) in \( X \) is the recipient of the maximal-provenance coin in \( X \) with coin object \( y \).

The key extension in sovereign cryptocurrencies over NFTs is the notion of coin redemption. Informally, a redemption claim by \( p \) against \( q \) is a \( q \)-coin transfer from \( p \) to \( q \), with a request to transfer in return an \( R \)-coin, for some \( R \subseteq \Pi \). Such a redemption claim can be settled by \( q \) transferring back to \( p \) an \( R \)-coin, if \( q \) has one, else a \( p \)-coin, if \( q \) has one. More formally:

Definition 7 (Redemption Claim, Settled/Outstanding). Given agents \( P \subseteq \Pi \), a redemption claim \( x = (x' \xrightarrow{\text{redeem}(R)} q)_p \) by \( p \) against \( q \), \( p,q \in P \), is a \( q \)-coin transfer from \( p \) to \( q \) with metadata \( \text{redeem}(R) \), \( p \notin R \subseteq P \). Given a configuration \( c \in C^\ast(P)^\ast \), such a redemption claim \( x \in c \) is valid if \( p \) is the holder of the \( q \)-coin \( x' \) in \( c \). A coin transfer \( (y \xrightarrow{p} q)_q \) in \( c \), with \( q \) being the holder of \( y \) in \( c \), settles the redemption claim \( x \in c \) if either:
1. $y$ is an $r$-coin, $r \in R$, fresh if $r = q$, or
2. $y$ is a $p$-coin and $q$ does not hold any $R$-coin NFTs in $c$.

Given a configuration $c$ and a valid redemption claim $x \in c$, if $c$ has a coin transfer that settles $x$ then $x$ is settled in $c$, else $x$ is outstanding in $c$.

Note how a redemption claim can be used. (i) **Reduce exposure:** If $R = \emptyset$ then the redemption claim is a request to reduce the exposure of $p$ to $q$ by decreasing their mutual credit line, which is fulfilled upon settlement. (ii) **Reset provenance:** If $R = \{q\}$ then it is a request to exchange the $q$-coin $x'$ for a fresh $q$-coin, which can be settled promptly by $q$. (iii) **Purchase coin:** If $q \notin R \neq \emptyset$, then it is a request to purchase from $q$ an $R$-coin that $q$ holds, if any, using a $q$-coin that $p$ holds. If $q$ has no $R$ coins but has a $p$-coin, then it settles the claim by returning to $p$ a $p$-coin instead, reducing their mutual credit line.

The informal description of $\mathcal{SC}$ is thus: $\mathcal{SC}$ is like $\mathcal{NT}$, except that local states include coin NFTs and an agent must settle promptly any valid redemption claim against it. The description begs the question of time and causality: What does ‘promptly’ mean? There are at least two interpretations: synchronous and asynchronous. We present them in turn, noting upfront that the synchronous system is simpler to specify but does not afford a grassroots implementation, and hence will not be pursued.

**Synchronous Sovereign Personal Cryptocurrencies.** The following definition employs the notion of a synchronous distributed multiagent transition system [31].

**Definition 8 (SCs: Synchronous Sovereign Personal Cryptocurrencies).** The synchronous sovereign personal cryptocurrencies protocol $\mathcal{SC}s$ is the subset of $\mathcal{NT}$ with configurations over coins $C^*$ and transitions $TT$ over coin configurations such that for each $p$-transition $c \rightarrow c' \in TT_p$, $c'_p = c_p \cdot y$, if $c$ has a valid redemption claim $x$ against $p$ then $y$ settles some valid redemption claim in $c$ against $p$.

Note that once a configuration has unsettled valid redemption claims against $p$, then until $p$ settles all of them, any subsequent configuration will also have unsettled valid redemption claims against $p$, meaning that a correct agent $p$ must settle all valid outstanding redemption claims against it before it can do anything else.

The protocol $\mathcal{SC}s$ is synchronous as an agent cannot ignore a redemption claim by another agent once issued. This requirement can be satisfied if all acts of all agents are totally-ordered, as in a blockchain. But, the whole point of our effort is to build a foundation for a grassroots digital economy that avoids the cost of achieving consensus on a total ordering, as in blockchain consensus or Byzantine atomic broadcast. Hence, given this goal, we aim for asynchronous sovereign cryptocurrencies.

**Asynchronous Sovereign Personal Cryptocurrencies.** To specify asynchronous personal cryptocurrencies, we provide a causal—or Einsteinian ‘light-cone’—answer to the meaning of ‘promptly’: If a coin transfer by $p$ is causally
preceded by a redemption claim against \( p \), then the coin transfer must be a response to an outstanding redemption claim. In other words, in an asynchronous realization of redemption claims, a correct agent must settle redemption claims against it before it can perform a transfer that is preceded by such redemption claims.

The following causal precedence relation \( \succ_{SC} \) on \( SC \) configurations is the algebraic counterpart Lamport’s notion of ‘happened before’ [20].

**Definition 9 (Causal Precedence of Coins \( \succ_{SC} \)).** Given an \( SC \) configuration \( c \), we define a causal precedence relation \( \succ_{SC} \) on the coins in \( c \) to be the minimal strict partial order satisfying:

1. **Same Agent:** Coin transfers in the local state of an agent are ordered by \( \succ_{SC} \) according to their sequence order. For any \( p \in \Pi \) and any two coins \( x, x' \in c_p \), if \( x \) precedes \( x' \) in \( c_p \) then \( x \succ_{SC} x' \).
2. **Same Coin Object:** Any two coin transfers with the same coin object are ordered according to their provenance. Namely, for any two coins \( x, x' \in c \) with the same coin object and with provenances \( v, v' \) respectively, if \( v \) is a prefix of \( v' \) then \( x \succ_{SC} x' \).

For any \( x, x' \in c \), we say that \( x \) precedes \( x' \) in \( c \), equivalently that \( x \succ_{SC} x' \).

Note that \( \succ_{SC} \) may fail to order coins that form an equivocation, as there is no prefix relation among their provenances. The notion of causal precedence \( \succ_{SC} \) is algebraic, rather than operational, as it relates to a structure created during the run, rather than to the run itself. We employ it to define the conditions for when an agent dodges a redemption claim under the asynchronous interpretation of ‘promptly’.

**Definition 10 (Dodge Redemption Claims).** Let \( c \in C^*(P)^P \) be a coin configuration. If \( c \) has a valid redemption claim against \( p \) that precedes a coin transfer \( y \in c \) from \( p \), and \( y \) does not settle a redemption claim in \( c \), then \( y \) dodges redemption claims in \( c \) against \( p \) and, more generally, \( c \) dodges redemption claims.

With this we can define \( SC \).

**Definition 11 (SC: Sovereign Personal Cryptocurrencies).** The sovereign personal cryptocurrencies protocol \( SC \) is the subset of \( NT \) with configurations over coins \( C^* \) and with all \( p \)-transitions, \( p \in P \), over coin configurations \( c \rightarrow c' \in TT_p \), \( c'_p = c_p \cdot y \), in which \( y \) does not dodge redemption claims in \( c' \).

Namely, \( p \) cannot transfer any coin \( x \) it holds that is preceded by redemption claims against \( p \), before settling all these redemption claims. Note that the only change from the synchronous protocol is the added condition \( x \succ_{SC} y \).

As agents communicate with each other, the longer a redemption claim remains unsettled, the more coins in the system it would precede, and eventually all coins transferred to \( p \) would thus be ‘contaminated’ and could not be further transferred by \( p \) as long as \( p \) does not settle the outstanding redemption
claim, hampering $p$’s liquidity. Practically, we expect the situation of $p$ to worsen much quicker: If other agents become aware that $p$ is not responding to redemption claims as fast as possible, a ‘run on the bank’ would ensue against $p$, with all agents attempting to redeem their $p$-coins against any non-$p$ coins that $p$ holds. This would cause $p$ a severe loss of liquidity and possibly even insolvency. Hence, we expect agents to settle redemption claims as soon as they know of them, rather than as late as the protocol allows them to.

**Proposition 2.** SC is asynchronous.

A grassroots protocol [31] can have multiple disjoint instances deployed independently at different locations and over time, and interoperate once their networks are connected. This is precisely the property we need to support a grassroots cryptoeconomy. We prove the following:

**Theorem 2.** SC is grassroots.

### 4 Implementation

Here we present an implementation of the sovereign cryptocurrencies protocol SC by the coin dissemination protocol CD. The implementing protocol is still quite abstract, but is readily amenable to a peer-to-peer messaging implementation [30]. The two implementations can be composed into a messaging implementation of SC.

**Definition 12 (Block).** Given a function $X$ that maps agents $P \subseteq \Pi$ to payloads $X(P)$, a block over $P$ and $X$ is a triple $(p, i, x) \in P \times N \times X(P)$. Such a block is referred to as an $i$-indexed $p$-block with payload $x$. The set of blocks over $P$ and $X$ is denoted by $B_X(P)$.

Note that in the implementation of the coin dissemination protocol CD, a block’s payload is a coin. But first we recall the more general grassroots asynchronous block dissemination protocol ABD [31].

**Definition 13 (ABD: Asynchronous Block Dissemination).** Given $X$ that maps a set of agents to payloads, the asynchronous block dissemination protocol ABD has for each $P \subseteq \Pi$ the transition system $ABD = (P, B_X(P), c_0, T, \lambda)$, an empty set as the initial local state $c_0 = \{\emptyset\}^P$, transitions $T$ with a $p$-transition $c \rightarrow c' \in T_p$, $c'_p = c_p \cup \{b\}$, $b = (p', i, x)$, for every $p, p' \in P$, $i \in \mathbb{N}$, $x \in X(P)$, and either:

1. $p$-Creates-$b$: $p' = p$, $i = \max \{j : (p, j, x) \in c_p\} + 1$, or
2. $p$-Receives-$b$: $p' \neq p$, $b \in c_q \setminus c_p$ for some $q \in P$, provided $c_p$ has an $(i-1)$-indexed $p'$-block or $i = 1$.

The liveness condition $\lambda$ places transitions with the same label in the same set.

In other words, every agent $p$ can either add a consecutively-indexed $p$-block to its local state or obtain from another agent a block it does not have, provided it already has its preceding block.
Definition 14 (ABD Complete and Consistent Configurations). An ABD configuration \( c \in B_X(P) \) is complete if for every \( p \)-block \((p, i, x)\) in \( c_q \), \( p, q \in P \), \( c_q \) has an \( i' \)-indexed \( p \)-block for every \( 1 \leq i' < i \). It is consistent if \((p, i, x), (p, i, x') \in c \) implies that \( x = x' \), and if \((p, i, x) \in c_q \) then \((p, i, x) \in c_p \).

Observation 2 A correct ABD run has complete and consistent configurations.

Theorem 3. ABD can implement SC.

Proof (outline of Theorem 3). We show that SC (Def. 11) is monotonically-complete wrt the prefix relation \( \preceq_{SC} \) (Def. 13, Prop. 4). We define the coin dissemination protocol \( CD \) (Def. 19) using a causal precedence relation \( \succ_{CD} \) on coin block occurrences (Def. 17) to be the subset of the asynchronous block dissemination protocol ABD (Def. 13) in which block payloads are coins (Def. 5) and transitions create configurations that are complete and consistent wrt to their coins (Def. 2), and do not dodge redemption claims (Def. 7). Next we define an implementation \( \sigma \) (Def. 21) on CD configurations that maps each local CD \( p \)-state, namely a set of coin blocks, to a local SC \( p \)-state with a sequence of coins, and show that CD is monotonically-complete wrt \( \preceq_{CD} \) (Prop. 4). We then present an inverse \( \hat{\sigma} \) (Def 22) to \( \sigma \) and use it prove that \( \sigma \) is order preserving wrt \( \preceq_{CD} \) and \( \preceq_{SC} \) and is productive (Prop. 5). Hence, according to Theorem 1 of [31], \( \sigma \) is a correct and complete implementation of SC by CD, which completes the proof.

The implementation ensures that a block known to a correct agent will be known eventually to all correct agents, and that if an equivocator attacks correct miners then eventually all correct miners will know of the equivocator. This is elaborated in Appendix 8. Recall that the exclusion of equivocations in a sovereign currency is achieved by the sovereign via coin redemption.

5 Conclusions

We have presented sovereign cryptocurrencies as foundation for a grassroots cryptoeconomy, discussed their possible uses, presented their implementation in terms of standard distributed computing protocols, and proved the implementations correct and equivocation-resilient. Much remains to be done, including analytic, computational and experimental analysis, as well as proof-of-concept implementation and deployment.

Acknowledgements I thank Nimrod Talmon, Gal Shahaf, Ouri Poupko, Michael Warner and Matt Prewitt for discussions and feedback. Ehud Shapiro is the Incumbent of The Harry Winreb Professorial Chair of Computer Science and Biology at Weizmann. Part of this work was carried out while I was a visiting scholar at Columbia University.
References

1. Christopher Allen. The Path to Self-Sovereign Identity, 2016, http://www.lifewithalacrity.com/2016/04/the-path-to-self-sovereign-identity.html. [accessed 2022].
2. Yoni Assia and Omri Ross. Good dollar experiment. Retrieved 2022 from https://www.gooddollar.org/wp-content/uploads/2018/11/GD-Wealth-Distribution-Position-Paper.pdf.
3. Alex Auvolat, Davide Frey, Michel Raynal, and François Taïani. Money transfer made simple: a specification, a generic algorithm, and its proof, 2021.
4. Gabriel Bracha. Asynchronous byzantine agreement protocols. Information and Computation, 75(2):130–143, 1987.
5. Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. page 173–186, New Orleans, Louisiana, USA, 1999. Proceedings of the third symposium on Operating systems design and implementation, USENIX Association.
6. Circles. Circles: A decentralised universal basic income platform based on personal currencies, Retrieved 2021 https://joincircles.net/.
7. Daniel Collins, Rachid Guerraoui, Jovan Komatovic, Matteo Monti, Athanasios Xygkis, Matej Pavlovic, Petr Kuznetsov, Yvonne-Anne Pignolet, Dragos-Adrian Seredinschi, and Andrei Tonkikh. Online payments by merely broadcasting messages (extended version). arXiv preprint arXiv:2004.13184, 2020.
8. Pranav Dandekar, Ashish Goel, Ramesh Govindan, and Ian Post. Liquidity in credit networks: A little trust goes a long way. In Proceedings of the 12th ACM conference on Electronic commerce, pages 147–156, 2011.
9. Sourav Das, Zhuolun Xiang, and Ling Ren. Asynchronous data dissemination and its applications. In Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, pages 2705–2721, 2021.
10. Bitcoin Developer. P2p network, Retrieved 2022.
11. Ryan Fugger. Money as ious in social trust networks & a proposal for a decentralized currency network protocol. Hypertext document. Available electronically at http://ripple. sourceforge. net, 106, 2004.
12. Ashish Goel, Sanjeev Khanna, Sharath Raghvendra, and Hongyang Zhang. Connectivity in random forests and credit networks. In Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms, pages 2037–2048. SIAM, 2014.
13. Rachid Guerraoui, Petr Kuznetsov, Matteo Monti, Matej Pavlovic, and Dragos-Adrian Seredinschi. At2: asynchronous trustworthy transfers. arXiv preprint arXiv:1812.10844, 2018.
14. Rachid Guerraoui, Petr Kuznetsov, Matteo Monti, Matej Pavlovic, and Dragos-Adrian Seredinschi. The consensus number of a cryptocurrency (extended version). Distributed Computing, pages 1–15, 2021.
15. Heiko Hees, Gustav Friis, Kristoffer Naerland, Aleexa Howitt, Tatu Kärki, and Andreas Fletcher. Trustlines network white paper, 2021, https://docs.trustlines.network/assets/pdf/Trustlines_Network_Whitepaper_2021.pdf.
16. A Howitt. Roadmap to a government-independent basic income (ubi) digital currency, 2019.
17. Idit Keidar, Eleftherios Kokoris-Kogias, Oded Naor, and Alexander Spiegelman. All you need is dag, 2021.
18. Idit Keidar, Oded Naor, and Ehud Shapiro. Cordial miners: A family of simple, efficient and self-contained consensus protocols for every eventuality. arXiv preprint arXiv:2205.09174, 2022.
19. Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Annual international cryptology conference*, pages 357–388. Springer, 2017.

20. Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications*, 1978.

21. Hyman P Minsky and Henry Kaufman. *Stabilizing an unstable economy*, volume 1. McGraw-Hill New York, 2008.

22. Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2019.

23. Ouri Poupko, Gal Shahaf, Ehud Shapiro, and Nimrod Talmon. Building a sybil-resilient digital community utilizing trust-graph connectivity. *IEEE/ACM transactions on networking*, 29(5):2215–2227, 2021.

24. Ouri Poupko, Ehud Shapiro, and Nimrod Talmon. Sybil-resilient coin minting. 2020.

25. Promify. Promify white paper, Retrieved 2022 from https://promify.io/whitepaper.

26. Simon De La Rouviere. In the future, everyone will have their own cryptocurrency. Retrieved 2022 from https://simondlr.tumblr.com/post/70089813484/in-the-future-everyone-will-have-their-own.

27. Gal Shahaf, Ehud Shapiro, and Nimrod Talmon. Sybil-resilient reality-aware social choice. Macao, China, 2019. the 28th International Joint Conference on Artificial Intelligence.

28. Gal Shahaf, Ehud Shapiro, and Nimrod Talmon. Genuine personal identifiers and mutual sureties for sybil-resilient community growth. In *International Conference on Social Informatics*, pages 320–332. Springer, 2020.

29. Gal Shahaf, Ehud Shapiro, and Nimrod Talmon. Egalitarian and just digital currency networks. In *Proceedings of AAMAS ’21*, pages 1649–1651, 2021.

30. Ehud Shapiro. The blocklace: A partially-ordered generalization of the blockchain and its grassroots consensus protocol stack. *To appear. Also accessible as Section 4 of an earlier version of this paper: https://arxiv.org/abs/2112.13650v8*, 2022.

31. Ehud Shapiro. Multiagent transition systems with safety and liveness faults: A compositional foundation for fault-resilient distributed computing. *arXiv preprint arXiv:2112.13650*, 2022.

32. Robert Shostak, Marshall Pease, and Leslie Lamport. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, 1982.

33. Paulo Silva, Miguel Matos, and João Barreto. Nimblechain: Low-latency consensusless cryptocurrencies in general-purpose permissionless blockchains. *arXiv preprint arXiv:2108.12387*, 2021.

34. Joseph Stiglitz, Joseph Eugene Stiglitz Stiglitz, and Bruce Greenwald. *Towards a new paradigm in monetary economics*. Cambridge university press, 2003.

35. Nguyen Tran, Jinyang Li, Lakshminarayanan Subramanian, and Sherman S. M. Chow. Optimal sybil-resilient node admission control. In *2011 Proceedings IEEE INFOCOM*, pages 3218–3226, 2011.

36. Bimal Viswanath, Ansley Post, Krishna P. Gummadi, and Alan Mislove. An analysis of social network-based sybil defenses. *SIGCOMM Comput. Commun. Rev.*, 40(4):363–374, 2010.

37. W3C. Decentralized identifiers (did) v1.0, Retrieved 2022 from https://www.w3.org/TR/did-core/.
38. Fennie Wang and Primavera De Filippi. Self-sovereign identity in a globalized world: Credentials-based identity systems as a driver for economic inclusion. *Frontiers in Blockchain*, 2:28, 2020.

39. Qin Wang, Rujia Li, Qi Wang, and Shiping Chen. Non-fungible token (nft): Overview, evaluation, opportunities and challenges. *arXiv preprint arXiv:2105.07447*, 2021.

40. Maofan Yin, Dahlia Malkhi, Michael K Reiter, Guy Golan Gueta, and Ittai Abraham. Hotstuff: Bft consensus with linearity and responsiveness. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, pages 347–356, 2019.
6 Proofs

Proof (of Observation 1). Informally, the two transitions of NT satisfy the asynchrony condition in the definition of asynchronous transition systems [31]: A \( p \)-Creates transition does not depend on the state of agents other than \( p \). A \( p \)-Transfers-\( y \) transition is enabled once \( p \) is the holder of \( y \), and is only disabled if \( p \) takes this transition, namely transfers \( y \) to another agent. \( \square \)

Proof (of Proposition 1). Let \( r = c_0 \rightarrow c_1 \rightarrow \ldots \) be a correct run of NT over \( P \subseteq \Pi \). We prove by induction on the index of the configurations of \( r \). The initial configuration \( c_0 \) is empty and therefore vacuously consistent and complete. Assume that \( c \in r \) is consistent and complete, and consider a \( p \)-transition \( c \rightarrow c' \in TT_p \), \( c'_p = c_p \cdot y \), \( y = (x \overset{\delta_p}{\rightarrow} q)_p \), \( p, q \in P \). The new configuration \( c' \) is consistent since \( c \) is consistent by the inductive assumption, and the set of NFTs of \( p \) following a \( p \)-transition is consistent by the definition of transition. And \( c' \) is complete since \( c \) is complete by the inductive assumption and \( y \), the only new NFT in \( c' \), extends the provenance of \( x \) by one agent, \( q \). \( \square \)

Proof (Proof of Theorem 1). We show that \( NT \) is a non-interfering, interactive, asynchronous protocol, which, according to Theorem 3 of [31] is a sufficient condition for \( NT \) to be grassroots.

Observation 1 established that \( NT \) is asynchronous. Let \( NT \) be the member of \( NT \) over \( P \), and \( c \) an \( NT \) configuration.

Regarding non-interference, if a \( p \)-transition of \( NT \) can be taken during a computation by some subset \( P' \) of \( P \), with \( p \in P' \), it can still be taken by \( p \) in a \( NT \) configuration in which the members of \( P \setminus P' \) are in their initial state, namely have not yet created or transferred any NFT, as, by this assumption, the \( p \)-transition is either an initial NFT, which can be taken in any state, or a transfer NFT of some payload involving only members of \( P' \), and hence agnostic to the presence of members of \( P \setminus P' \). Hence \( NT \) is non-interfering.

Regarding interactivity, when two disjoint instances \( NT(P_1) \) and \( NT(P_2) \) are combined, \( P_1 \cap P_2 = \emptyset \), then \( NT(P_1 \cup P_2) \) includes transfers of NFTs from members of \( P_1 \) to members of \( P_2 \) and vice versa, which are not available in \( NT(P_1) \cup NT(P_2) \). Hence \( NT \) is interactive. This completes the proof. \( \square \)

Proof (of Proposition 2). The argument for the asynchrony of \( NT \) (Observation 1) holds as is for coin creation and transfer in the absence of redemption claims. We argue that redemption claims and their settlement do not hamper asynchrony. A redemption claim by another agent cannot disable a Create transition. Regarding a Transfer transition, if a configuration \( c \) has no redemption claim against \( p \) that precedes a coin \( x \) held by \( p \) in \( c \), then no existing or subsequent redemption claim against \( p \) will precede \( x \) in any subsequent configuration \( c' \), by the definition of the causal precedence relation. Hence, a transfer of \( x \) enabled in \( c \) will also be enabled in any subsequent configuration \( c' \) for which \( c_p = c'_p \), and hence \( SC \) satisfies the asynchrony condition wrt the prefix relation over complete and consistent configurations. \( \square \)
Proof (outline of Theorem 2). The proof is essentially the same as the proof of Theorem 1 that, together with Proposition 2, show that SC is a non-interfering, interactive, asynchronous protocol, which a sufficient condition for being grassroots.

Proof (outline of Theorem 3). We show that SC (Def. 11) is monotonically-complete wrt the prefix relation \( \preceq_{SC} \) (Def. 13, Prop. 3). We define the coin dissemination protocol \( CD \) (Def. 19) using a causal precedence relation \( \succ_{CD} \) on coin block occurrences (Def. 17) to be the subset of the asynchronous block dissemination protocol \( ABD \) (Def. 13) in which block payloads are coins (Def. 5) and transitions create configurations that are complete and consistent wrt to their coins (Def. 2), and do not dodge redemption claims (Def. 7). Next we define an implementation \( \sigma \) (Def. 21) on \( CD \) configurations that maps each local \( CD \) p-state, namely a set of coin blocks, to a local \( SC \) p-state with a sequence of coins, and show that \( CD \) is monotonically-complete wrt \( \preceq_{CD} \) (Proposition 4).

We then show that \( \sigma \) is order preserving wrt \( \preceq_{CD} \) and \( \preceq_{SC} \) and is productive (Proposition 5). Hence, according to Theorem 1 of [31], \( \sigma \) is a correct and complete implementation of \( SC \) by \( CD \), which completes the proof.

As in \( ABD \) copies of the same block may occur in the local states of different agents, its causal precedence relation is among block occurrences, or copies, not blocks.

Definition 15 (Causal Precedence on Block Occurrences \( \succ_{ABD} \)). Given a complete and consistent configuration \( c \in B_X(P)^P \), the causal precedence relation on block occurrences \( \succ_{ABD} \) is the minimal strict partial order on block occurrences in \( c \) that includes \( b \succ_{ABD} b' \) if:

1. \textit{Same agent}: \( p = p', i < i', \) or
2. \textit{Same block}: \( b \) and \( b' \) are two copies of the same block, \( b \) occurs in \( c_p \), \( b' \) occurs in \( c_q, q \neq p \).

Definition 16 (\( \preceq_{SC} \)). The partial order \( \preceq_{SC} \) is defined over \( SC \) configurations by \( c \preceq_{SC} c' \) for \( c, c' \in C^*(P)^P \), if \( c \) and \( c' \) are consistent and complete and \( c_p \) is a prefix of \( c'_p \) for every \( p \in P \).

Proposition 3. \( SC \) is monotonically-complete with respect to \( \preceq_{SC} \).

Proof (of Proposition 3). A transition system \( SC \in ST \) over \( P \subseteq \Pi \) is monotonic wrt \( \preceq_{SC} \) since according to Proposition 1 all configurations in every run of \( NT \) are consistent and complete, and being a subset of \( NT \) over \( P \), \( SC \) computations are also \( NT \) computations and, hence all configurations in every run of \( SC \) are also consistent and complete, and the result of a \( p \)-transition \( c \rightarrow c' \in TT \) is that \( c_p \) is a prefix of \( c'_p \) with other local states intact, implying \( c \preceq_{SC} c' \).

To prove that \( SC \) is monotonically-complete, we show that it is \( \epsilon \)-monotonically-complete. First, as \( \preceq_{SC} \) is defined via the prefix relation, infinite ascending chains in it are unbounded, as required. Second, consider two configurations \( c \preceq_{SC} c'' \), and let \( x \in C(P) \) be a \( p \)-coin in \( c'' \setminus c \), \( p \in P \) that is not causally preceded by any other coin in \( c' \setminus c \). If \( x \) is a \( p \)-coin object then the \( p \)-creates-\( x \) transition...
\( c \rightarrow c' \in TT \). If \( x \) is a \( p \)-coin transfer with payload \( x' \), then by assumption that \( c'' \) is consistent and complete \( x \in c'' \), and by the choice of \( x \) as not causally preceded by any other coin in \( c' \setminus c, x' \) must be in \( c \). Hence the \( p \)-Transfers-\( x \) transition \( c \rightarrow c' \in TT \). In both cases, \( c' \) is complete and consistent since \( c \) is, \( c \preceq_{SC} c' \), and \( c' \preceq_{SC} c'' \) since \( c'_p \) is a prefix of \( c''_p \) and \( c'_q = c''_q \) for every \( q \neq p \in P \), both by construction. Hence \( SC \) is \( \varepsilon \)-monotonically-complete wrt \( \preceq_{SC} \).

Note that a configuration \( c \in B_C(P)^P \), \( P \subseteq \Pi \), has for each agent \( p \in P \) a local state \( c_p \) that includes a set of blocks \( c_p \subseteq B_C(P) \), where the payload \( x \) of each block \( (p', i, x) \in c_p \) is a coin \( x \in C(P) \) over the agents \( P \).

**Definition 17 (Causal Precedence of Coin Block Occurrences \( \succ_{CD} \)).**

Given a configurations \( c \in B_C(P)^P \), the causal precedence relation on coin block occurrences \( \preceq_{CD} \) is the union of the causal precedence relations on coins \( \succ_{SC} \) and on block occurrences \( \succ_{ABD} \), namely for two block occurrences \( b, b' \in c \), \( b \succ_{CD} b' \) \( b = (p, i, x) \), \( b' = (p', i', x') \) if \( b \succ_{ABD} b' \) or \( x \succ_{SC} x' \).

Next we now define the coin dissemination protocol \( CD \). Its main difference from the sovereign cryptocurrencies protocol \( SC \) is that in the latter coin transfer transitions and their associated redemption conditions refer to the entire configuration, whereas in \( CD \) they refer to the local set of coin blocks of the agent making the transition. Local states are updated also by the transition \( p\text{-Receives-}b \), in which \( p \) adds to its local state the block \( b \) transferred to it by another agent. Hence, we define for each agent \( p \) its local view of the coins in \( c \).

**Definition 18 (Coin view \( \gamma \)).** Given a set of coin blocks \( B \subseteq B_C(P) \), the coin view \( \gamma(B) \) of \( B \) is the \( SC \) configuration \( c \in C^*(P)^P \), defined to have for each coin block \( (p, i, x) \in B \) the coin \( x \) in position \( i \) in the sequence of coins \( c_p \), provided the resulting \( c \) is a well-defined, complete and consistent \( SC \) configuration over \( P \).

Namely, \( \gamma(B) \) is undefined if \( B \) has a missing block index, an incomplete coin history, or an equivocation. The coin dissemination protocol \( CD \) is the subset of the asynchronous block dissemination protocol \( ABD \) in which states are restricted to coin blocks and correct transitions over coin blocks are further restricted so that the coin views of agents preserve completeness, consistency and do not dodge redemption claims. More formally,

**Definition 19 (\( CD \): Coin Dissemination).** The coin dissemination protocol \( CD \) is the subset of \( ABD \) where:

1. **States:** \( CD \) local states are \( ABD \) local states in which block payloads are coins, namely \( X = C \), and

2. **Transitions:** Correct \( CD \) transitions are correct \( ABD \) transitions \( c \rightarrow c' \) among \( CD \) configurations over \( P \), satisfying that if \( \gamma(c_p) \) is a complete and consistent \( SC \) configuration that does not dodge redemption claims, then so is \( \gamma(c'_p) \), for every \( p \in P \).
Definition 20 ($\preceq_{CD}$). The partial order $\preceq_{CD}$ is defined over $CD$ configurations by $c \preceq_{CD} c'$ for $c, c' \in Bc(P)^P$, if $\gamma(c_p) \preceq_{SC} \gamma(c'_p)$ for every $p \in P$.

Proposition 4. $CD$ is monotonically-complete with respect to $\preceq_{CD}$.

Proof. Let $CD = (P, Bc(P), c0, T, \lambda) \in CD$. Given a $CD$ $p$-transition $c \rightarrow c' \in T_p$ that adds the $p$-coin-block $b$ to $c_p$, then $c_p \subseteq c'_p \cup \{b\} = c'_q$, and $\forall q \neq p : c_q = c'_q$, hence $c \preceq_{CD} c'$ and $CD$ is monotonic wrt $\preceq_{CD}$.

To prove that $CD$ is monotonically-complete, we show that it is $\epsilon$-monotonically-complete. First, as $\preceq_{CD}$ is defined via the prefix relation on $SC$, infinite ascending chains are unbounded, as required. Second, consider the $CD$ transition system $CD = (P, Bc(P), c0, T, \lambda)$, and the two configurations of it $c \preceq_{CD} c''$, let $b \in Bc(P)$ be a coin-block in $c'' \setminus c$ that is not causally preceded by any other coin-block in $c'' \setminus c$, $b \in c''_p$, and define $c'$ by $c'_p := c_p \cup \{b\}$, $c'_q := c_q$ for every $q \neq p \in P$. We consider several cases for $b$:

1. $b \in c''_p$ is an occurrence of a $q$-block with an occurrence $b' \in c'_q, q \neq p \in P$.
   In which case $b$ is casually preceded by $b'$, thus by assumption $b'$ occurs in $c_q$, and hence $c \rightarrow c' \in T$.

2. $b = (p, i, x) \in c''_p$. If $i > i$ then by assumptions (that $\gamma(c_p)$ is complete and consistent, and that $b$ is not causally preceded by any block in $c'' \setminus c$) the preceding $(i-1)$-indexed $p$-block is in $c_p$. There are two cases:
   (a) $x$ is a $p$-coin object.
   (b) $x$ is a $p$-coin transfer of payload $x'$. By completeness of $c'$, a block $b'$ with the coin $x'$ as payload occurs in $c'_p$. By assumption that $b$ is not preceded in $c'' \setminus c, b' \in c_p$.

   In both cases $c \rightarrow c' \in T$ as $\gamma(c'_p)$ is complete, consistent, and does not dodge redemption claims.

In all cases, $\gamma(c'_p)$ is complete, consistent, does not dodge redemption claims, $c \prec_{CD} c'$, and $c' \preceq_{CD} c''$ since $\gamma(c'_p)$ is a prefix of $\gamma(c''_p)$ and $c'_q = c''_q$ for every $q \neq p \in P$, both by construction. Hence $CD$ is $\epsilon$-monotonically-complete and thus monotonically-complete wrt $\preceq_{CD}$. \hfill \Box

Definition 21 ($\sigma$). The local implementation $\sigma : Bc(P)^P \rightarrow C^*(P)^P$ is defined as follows. For a configuration $c \in Bc(P)^P$ and $p \in P$, let $c_p/p$ be the set of $p$-blocks in $c_p$. If $c_p/p$ is a set of consecutively indexed coin blocks $\{(p, 1, x_1), (p, 2, x_2), \ldots, (p, n, x_n)\}$ then $\sigma(c_p) := x_1, x_2, \ldots, x_n$, else $\sigma(c_p)$ is undefined.

Proposition 5. $\sigma$ is order-preserving wrt $\preceq_{CD}$ and $\preceq_{SC}$

According to Observation 2 of [31], identifying an inverse $\sigma$ to $\sigma$ greatly simplifies the proof.

Definition 22 ($\hat{\sigma}$). Given an $SC$ configuration $c \in C^*(P)^P$, $\hat{\sigma}(c) \in Bc(P)^P$ converts the coins in $c$ to coin blocks and lets every agent have a copy of every coin block. To define $\hat{\sigma}$ we first define $\hat{\gamma}$. Given a configuration $c \in C^*(P)^P$, then for each coin $x$ with index $i$ in the sequence $c_p$, the block $(p, i, x) \in \hat{\gamma}(c)$. Then $\hat{\sigma}(c)_p := \hat{\gamma}(c)$ for every $p \in P$. 

23
Observation 3. Given \( c, c' \in C^*(P)^P \) such that \( c_0 \xrightarrow{*} c \) is an SC run and \( c \succeq_{SC} c' \), then \( \sigma(\tilde{c}(c)) = c \) and \( \sigma(c) \succeq_{CD} \tilde{c}(c') \).

Proof (Proof of Proposition 23). According to Observation 2 the function \( \tilde{c} \) satisfied the conditions of Observation 2 of \([31]\).

This completes the proof of Theorem 3.

7 Multiagent Transition Systems with Safety and Liveness Faults

Here we replicate definitions and results from \([31]\) used in this paper. For their explanations, examples and proofs please consult the original reference \([31]\).

Given a set \( S, S^* \) denotes the set of sequences over \( S, S^+ \) the set of nonempty sequences over \( S, \) and \( \Lambda \) the empty sequence. Given \( x, y \in S^* \), \( x \cdot y \) denotes the concatenation of \( x \) and \( y \), and \( x \preceq y \) denotes that \( x \) is a prefix of \( y \). Two sequences \( x, y \in S^* \) are consistent if \( x \preceq y \) or \( y \preceq x \), inconsistent otherwise.

Definition 23 (Transition System, Computation, Run). Given a set \( S \), referred to as states, the transitions over \( S \) are all pairs \( (s, s') \in S^2 \), also written \( s \rightarrow s' \). A transition system \( TS = (S, s_0, T, \lambda) \) consists of a set of states \( S \), an initial state \( s_0 \in S \), a set of correct transitions \( T \subseteq S^2 \), and a liveness condition \( \lambda \) which is a set of sets correct transitions; when \( \lambda \) is omitted the default liveness condition is \( \lambda = \{T\} \). A computation of \( TS \) is a sequence of transitions \( r = s \rightarrow s' \rightarrow \cdots \subseteq S^2 \). A run of \( TS \) is a computation that starts from \( s_0 \).

Definition 24 (Safe, Live and Correct Run). Given a transition system \( TS = (S, s_0, T, \lambda) \), a computation \( r \) is safe, also \( r \subseteq T \), if every transition of \( r \) is correct, and \( s \xrightarrow{r} s' \subseteq T \) denotes the existence of a safe computation (empty if \( s = s' \)) from \( s \) to \( s' \).

A transition \( s' \rightarrow s'' \in S^2 \) is enabled on \( s \) if \( s = s' \). A run is live wrt \( L \in \lambda \) if either \( r \) has a nonempty suffix in which no transition in \( L \) is enabled, or every suffix of \( r \) includes an \( L \) transition. A run \( r \) is live if it is live wrt every \( L \in \lambda \). A run \( r \) is correct if it is safe and live.

Definition 25 (Specification; Safe, Live, Correct and Complete Implementation). Given two transition systems \( TS = (S, s_0, T, \lambda) \) (the specification) and \( TS' = (S', s'_0, T', \lambda') \), an implementation of \( TS \) by \( TS' \) is a function \( \sigma : S' \rightarrow S \) where \( \sigma(s'_0) = s_0 \), in which case the pair \((TS', \sigma)\) is referred to as an implementation of \( TS \). Given a computation \( r' = s'_1 \rightarrow s'_2 \rightarrow \cdots \) of \( TS', \) \( \sigma(r') \) is the (possibly empty) computation \( \sigma(s'_1) \rightarrow \sigma(s'_2) \rightarrow \cdots \), with stutter transitions in which \( \sigma(s'_i) = \sigma(s'_{i+1}) \) removed. The implementation \((TS', \sigma)\) of \( TS \) is safe/live/correct if \( \sigma \) maps every safe/live/correct \( TS' \) run \( r' \) to a safe/live/correct \( TS \) run \( \sigma(r') \), respectively, and is complete if every correct run \( r \) of \( TS \) has a correct run \( r' \) of \( TS' \) such that \( \sigma(r') = r \).
Definition 26 (σ: Locally Safe, Productive, Locally Complete). Given two transition systems \( TS = (S, s_0, T, \lambda) \) and \( TS' = (S', s'_0, T', \lambda') \) and an implementation \( \sigma : S' \rightarrow S \). Then \( \sigma \) is:

1. **Locally Safe** if \( s'_0 \xrightarrow{\sigma} y_1 \rightarrow y_2 \subseteq T' \) implies that \( s_0 \xrightarrow{\sigma} x_1 \xrightarrow{\sigma} x_2 \subseteq T \) for \( x_1 = \sigma(x'_1) \) and \( x_2 = \sigma(x'_2) \) in \( S \). If \( x_1 = x_2 \) then the \( T' \) transition \( x'_1 \rightarrow x'_2 \) stutters \( T \).

2. **Productive** if for every \( L \in \lambda \) and every correct run \( r' \) of \( TS' \), either \( r' \) has a nonempty suffix \( r'' \) such that \( L \) is not enabled in \( \sigma(r'') \), or every suffix \( r'' \) of \( r' \) activates \( L \), namely \( \sigma(r'') \) has an \( L \)-transition.

3. **Locally Complete** if \( s'_0 \xrightarrow{\sigma} x_1 \rightarrow x_2 \subseteq T \), implies that \( s'_0 \xrightarrow{\sigma} x'_1 \xrightarrow{\sigma} x'_2 \subseteq T' \) for some \( x'_1, x'_2 \in S' \) such that \( x_1 = \sigma(x'_1) \) and \( x_2 = \sigma(x'_2) \).

Proposition 6 (σ Correct). If an implementation \( \sigma \) is locally safe and productive then it is correct, and if in addition it is locally complete then it is complete.

Definition 27 (Transition System Subset). Given a transition system \( TS = (S, s_0, T, \lambda) \), a transition system \( TS' = (S', s'_0, T', \lambda') \) is a **subset** of \( TS \), \( TS' \subseteq TS \), if \( s'_0 = s_0, S' \subseteq S, T' \subseteq T \), and \( \lambda' \) is \( \lambda \) restricted to \( T' \).

Definition 28 (Can Implement). Given transition systems \( TS = (S, s_0, T, \lambda) \), \( TS' = (S', s'_0, T', \lambda') \), \( TS' \can implement \ TS \) if there is a subset \( TS'' = (S'', s''_0, T'', \lambda'') \), \( TS'' \subseteq TS' \) and a correct and complete implementation \( \sigma : S'' \rightarrow S \) of \( TS \) by \( TS'' \).

Proposition 7 (Transitivity of Correct & Complete Implementations). The composition of safe/live/correct-complete implementations is safe/live/correct-complete, respectively.

Definition 29 (Partial Order). A reflexive partial order on a set \( S \) is denoted by \( \preceq_S \) (with \( S \) omitted if clear from the context), \( s \prec s' \) stands for \( s \preceq s' \) & \( s' \neq s \), and \( s \simeq s' \) for \( s \preceq s' \) & \( s' \preceq s \). The partial order is **strict** if \( s \simeq s' \) implies \( s = s' \) and **unbounded** if for every \( s \in S \) there is an \( s' \in S \) such that \( s \prec s' \). We say that \( s, s' \in S \) are **consistent** wrt \( \preceq \) if \( s \preceq s' \) or \( s' \preceq s \) (or both).

Definition 30 (Monotonic & Monotonically-Complete Transition System). Given a partial order \( \preceq \) on \( S \), a transition system \( TS = (S, s_0, T, \lambda) \) is **monotonic** with respect to \( \preceq \) if \( s \preceq s' \rightarrow s \preceq s' \) in \( T \). It is **monotonically-complete** wrt \( \preceq \) if, in addition, \( s_0 \xrightarrow{s} s \subseteq T \) and \( s \preceq s' \implies s \xrightarrow{s} s' \subseteq T \).

Definition 31 (ε-Monotonic Completeness). A transition system \( TS = (S, s_0, T, \lambda) \), monotonic wrt a partial order \( \preceq \) on \( S \), is **ε-monotonically-complete** wrt \( \preceq \) if infinite ascending chains in \( \preceq \) are unbounded and \( s \prec s' \) implies that there is a transition \( s \rightarrow s' \in T \) such that \( s \prec s' \) and \( s' \preceq s'' \).

Proposition 8 (ε-Monotonic Completeness). A transition system that is ε-monotonically-complete is monotonically-complete.
Definition 32 (Order-Preserving Implementation). Let transition systems 
\( TS = (S, s_0, T, \lambda) \) and \( TS' = (S', s'_0, T', \lambda') \) be monotonic wrt the partial orders \( \preceq \) and \( \preceq' \), respectively. Then an implementation \( \sigma : S' \rightarrow S \) of \( TS \) by \( TS' \) is order-preserving wrt \( \preceq \) and \( \preceq' \) if:

1. **Up condition:** \( y_1 \preceq' y_2 \) implies that \( \sigma(y_1) \preceq \sigma(y_2) \)
2. **Down condition:** \( s_0 \xrightarrow{*} x_1 \subseteq T, x_1 \preceq x_2 \) implies that there are \( y_1, y_2 \in S' \)
   such that \( x_1 = \sigma(y_1), x_2 = \sigma(y_2), s_0 \xrightarrow{*} y_1 \subseteq T' \) and \( y_1 \preceq' y_2 \).

Theorem 4 (Correct & Complete Implementation Among Monotonically-Complete Transition Systems). Assume two transition systems \( TS = (S, s_0, T, \lambda) \) and \( TS' = (S', s'_0, T', \lambda') \), monotonically-complete wrt the unbounded partial orders \( \preceq \) and \( \preceq' \), respectively, and an implementation \( \sigma : S' \rightarrow S \) by \( TS' \). If \( \sigma \) is order-preserving and productive then it is correct and complete.

Observation 4 (Representative Implementation State) Assume \( TS \) and \( TS' \) as in Theorem 4 and an implementation \( \sigma : S' \rightarrow S \) that satisfies the Up condition of Definition 32. If there is a function \( \hat{\sigma} : S \rightarrow S' \) such that \( x = \sigma(\hat{\sigma}(x)) \)
for every \( x \in S \), and \( x_1 \preceq x_2 \) implies that \( \hat{\sigma}(x_1) \preceq' \hat{\sigma}(x_2) \), then \( \sigma \) also satisfies the Down condition.

Proposition 9 (Restricting a Correct Implementation to a Subset). Let \( \sigma : C_2 \rightarrow S_1 \) be an order-preserving implementation of \( TS_1 = (S_1, s_1, T_1, \lambda_1) \) by \( TS_2 = (C_2, s_2, T_2, \lambda_2) \), monotonically-complete respectively with \( \preceq_1 \) and \( \preceq_2 \). Let \( TS'_1 = (S'_1, s_1, T_1', \lambda'_1) \subseteq TS_1 \) and \( TS'_2 = (C'_2, s_2, T_2', \lambda'_2) \subseteq TS_2 \) defined by \( C'_2 := \{ s \in C_2 : \sigma(s) \in S'_1 \} \), with \( T'_2 := T_2/C'_2 \), and assume that both subsets are also monotonically-complete wrt \( \preceq_1 \) and \( \preceq_2 \), respectively. If \( y_1 \rightarrow y_2 \in T'_2 \) & \( \sigma(y_1) \in S'_1 \) implies that \( \sigma(y_2) \in S'_1 \) then the restriction of \( \sigma \) to \( C'_2 \) is a correct and complete implementation of \( TS'_1 \) by \( TS'_2 \).

Definition 33 (Multiagent Transition System). Given agents \( P \subseteq \Pi \), a transition system \( TS = (C, c_0, T, \lambda) \), with configurations \( C \), initial configuration \( c_0 \), correct transitions \( T \subseteq C^2 \), and a liveness condition \( \lambda \) on \( T \), is multiagent over \( P \) if there is a multiagent partition \( C^2 = \bigcup_{p \in P} CC_p \) of \( C^2 \) into disjoint sets \( CC_p \) indexed by \( P \). \( CC_p \cap CC_q = \emptyset \) for every \( p \neq q \in P \). A transition \( t = s \rightarrow s' \in CC_p \) is referred to as a \( p \)-transition, and the set of correct \( p \)-transitions \( T_p \) is defined by \( T_p := T \cap CC_p \), for every \( p \in P \).

Definition 34 (Safe, Live & Correct Agents). Given a multiagent transition system \( TS = (C, c_0, T, \lambda) \) over \( P \) and a run \( r \) of \( TS \), an agent \( p \) is safe in \( r \) if \( r \) includes only correct \( p \)-transitions; is live in \( r \) if for every \( L \in \lambda \) for which \( L \subseteq T_p \), \( r \) is live wrt \( L \); and is correct in \( r \) if \( p \) is safe and live in \( r \).

Proposition 10. SCC over \( P \subseteq \Pi \) and \( S \) can implement any generic shared-memory multiagent transition system \( GS \) over \( P \) and \( S \).

Definition 35 (Centralized and Distributed Multiagent Transition System). A multiagent transition system \( TS = (C, c_0, T, \lambda) \) over \( P \) with multiagent partition \( C^2 = \bigcup_{p \in P} CC_p \) is distributed if:
1. \( C = S^P \) for some set \( S \), referred to as local states, namely each configuration \( c \in C \) consists of a set of local states in \( S \) indexed by \( P \), in which case we use \( c_p \in S \) to denote the local state of \( p \in P \) in configuration \( c \in C \), and
2. Any \( p \)-transition \( c \rightarrow c' \in CC_p \) satisfies that \( c'_p \neq c_p \) and \( c'_q = c_q \) for every \( q \neq p \in P \).

Else \( TS \) is centralized.

A partial order \( \leq \) over a set of local states \( S \) naturally extends to configurations \( C = S^P \) over \( P \subset \Pi \) and \( S \) by \( c \leq c' \) for \( c, c' \in C \) if \( c_p \leq c'_p \) for every \( p \in P \).

**Definition 36 (Distributed Transition System; Synchronous and Asynchronous).** Given agents \( P \subseteq \Pi \), local states \( S \), and a distributed transition system \( TS = (P, S, c_0, T, \lambda) \), then \( TS \) is asynchronous wrt a partial order \( \leq \) on \( S \) if:
1. \( TS \) is monotonic wrt \( \leq \), and
2. for every \( p \)-transition \( c \rightarrow c' \in T, T \) also includes the \( p \)-transition \( d \rightarrow d' \) for every \( d, d' \in C \) that satisfy the following asynchrony condition:
   \[
   c \preceq d, (c_p \rightarrow c'_p) = (d_p \rightarrow d'_p), \text{ and } d'_q = d_q \text{ for every } q \neq p \in P.
   \]

If no such partial order on \( S \) exists, then \( TS \) is synchronous.

**Definition 37 (Safety and Liveness Faults).** Given a transition system \( TS = (S, s_0, T, \lambda) \), a safety fault is a set of incorrect transitions \( F \subseteq S^2 \setminus T \).

A computation performs a safety fault \( F \) if it includes a transition from \( F \).

A liveness fault is a subset \( \lambda' \subseteq \lambda \) of the liveness condition \( \lambda \). An infinite run performs a liveness fault \( \lambda' \) if it is not live wrt \( L \) for some \( L \in \lambda' \).

**Definition 38 (Safety-Fault Resilience).** Given transition systems \( TS = (S, s_0, T, \lambda), TS' = (S', s'_0, T', \lambda') \) and a safety fault \( F \subseteq S'^2 \setminus T' \), a correct implementation \( \sigma : S' \rightarrow S \) is \( F \)-resilient if for any live \( TS' \) run \( r' \subseteq T \cup F \), the run \( \sigma(r') \) is correct.

**Definition 39 (Family of Multiagent Transition Systems; Protocol; Asynchronous; Subset).** Assume a function \( S \) that maps each set of agent \( P \subset \Pi \) into a set of local states \( S(P) \). A family \( F \) of multiagent transition systems over \( S \) is a set of transition systems such that for each set of agents \( P \subset \Pi \) there is one transition system \( TS(P) = (C(P), c_0(P), T(P), \lambda(P)) \) \( \in F \) with configurations \( C(P) \) over \( P \) and \( S(P) \), transitions \( T(P) \) over \( C(P) \) and a liveness condition \( \lambda(P) \). The states of \( F \) are \( S(F) := \bigcup_{P \subseteq \Pi} S(P) \). If a family of multiagent transition systems is distributed then we refer to it as a protocol, which is asynchronous if every transition system in \( F \) is asynchronous, and is a subset of protocol \( F' \) if for every \( P \subseteq \Pi \), \( TS(P) \) is a subset of \( TS'(P) \).

**Definition 40 (Projection of a Configuration).** Let \( P' \subset P \subset \Pi \) and let \( TS = (P, S, c_0, T) \) be a distributed transition system. The projection of a configuration \( c \in S^P \) on \( P' \), \( c/P' \), is the configuration \( c' \) over \( P' \) satisfying \( c'_p = c_p \) for all \( p \in P' \).
Definition 41 (Union of Distributed Transition Systems). Let $TS_1 = (P_1, S_1, c_0, T_1)$, $TS_2 = (P_2, S_2, c_0, T_2)$ be two distributed transition systems, $P_1 \cap P_2 = \emptyset$. Then the union of $TS_1$ and $TS_2$, $TS := TS_1 \cup TS_2$, is the multiagent transition systems $TS = (P_1 \cup P_2, S_1 \cup S_2, c_0, T)$ with initial state $c_0$ satisfying $c_0/P_1 = c_0$, $c_0/P_2 = c_0$, and all $p$-transitions $c \rightarrow c' \in T$, $p \in P$, satisfying $p \in P_1 \wedge (c/P_1 \rightarrow c'/P_1) \in T_1 \vee c/P_2 = c'/P_2$ or $p \in P_2 \wedge (c/P_2 \rightarrow c'/P_2) \in T_2 \vee c/P_1 = c'/P_1$.

Definition 42 (Grassroots). A protocol $\mathcal{F}$ supports grassroots composition, or is grassroots, if for every $\emptyset \subset P_1, P_2 \subset \Pi$ such that $P_1 \cap P_2 = \emptyset$, the following holds:

$$TS(P_1) \cup TS(P_2) \subset TS(P_1 \cup P_2)$$

Definition 43 (Interactive & Non-Interfering Protocol). A protocol $\mathcal{F}$ is interactive if for every $\emptyset \subset P_1, P_2 \subset \Pi$ such that $P_1 \cap P_2 = \emptyset$, the following holds: $TS(P_1 \cup P_2) \not\subseteq TS(P_1) \cup TS(P_2)$. It is non-interfering if for every $P' \subset P \subset \Pi$, with transition systems $TS = (P, S(P), c_0, T), TS' = (P', S(P'), c_0', T') \in \mathcal{F}$, and every transition $c_1' \rightarrow c_2' \in T'$, $T$ includes the transition $c_1 \rightarrow c_2$ for which $c_1' = c_1/P'$, $c_2' = c_2/P'$, and $c_1_p = c_2_p = c_0_p$ for every $p \in P \setminus P'$.

Theorem 5 (Grassroots Protocol). An asynchronous, interactive, and non-interfering protocol is grassroots.

8 Fault Resilience of ABD

Two properties of ABD (Def. 13) proved in [31] are important for our purpose: Fault-resilient dissemination and equivocation detection. We use ‘$p$ knows $b$’ in a run $r$ to mean that $b \in c_r$ for some $c \in r$.

Proposition 11 (ABD Block Liveness). In an ABD run, if a correct agent knows a block $b$ then eventually all correct agents know $b$.

Definition 44 (Equivocation). An equivocation by agent $p$ consists of two $p$-blocks $b = (p, i, s), b' = (p, i', s')$ where $i = i'$ but $s \neq s'$. An agent $p$ is an equivocator in $B$ if $B$ includes an equivocation by $p$. A set of blocks $B$ is equivocation-free if it does not include an equivocation.

The following corollary states that if an agent $p$ tries to mislead (e.g. double spend) correct agents by disseminating to different agents equivocating blocks, then eventually all correct agents will know that $p$ is an equivocator.

Corollary 1 (ABD Equivocation Detection). In an ABD run, if two blocks $b, b'$ of an equivocation by agent $p$ are each known by a different correct agent, then eventually all correct agents know that $p$ is an equivocator.