Akyba: Deterministic Cooperative Finance

on an Extended UTxO Ledger

Ange T. Yobo Ariady Putra

December 2025

Abstract

Informal savings and credit associations such as rotating savings (ROSCAs) and accumulating savings (ASCAs) coordinate capital without banks, but rely on social enforcement and informal record-keeping, limiting scale and creating dispute risk.

We present *Akyba*, a protocol that implements ROSCA and ASCA mechanisms as deterministic state machines under the extended UTxO (eUTxO) model. Group rules are enforced by transaction validation rather than trusted organizers, so participation parameters are fixed after initialization and accounting for contributions, distributions, and loan decisions is verifiable from the ledger.

The protocol uses a singleton state token, identity-linked reference tokens (CIP-68), and validator logic to enforce linear state evolution and conservation of value. We describe the state model, transaction structure, incentives, and security properties required to reproduce cooperative finance on-chain with predictable execution costs and without custodians.

1 Introduction

Rotating and accumulating savings associations have existed for centuries as a way to pool capital in environments without formal banking infrastructure. Participants commit to repeated contributions under fixed rules, enabling access to lump sums or credit through mutual coordination.

These systems function through social trust, reputation, and local enforcement. While effective in small groups, they do not scale beyond tight social networks and are vulnerable to organizer fraud, disputes, and record-keeping failures.

Blockchain systems enable coordination without trusted intermediaries. This paper describes a protocol that implements cooperative savings and credit as a deterministic state machine whose transitions are enforced by the ledger under the eUTxO model.

2 Design Goals

The protocol is constructed so that all group rules are enforced by transaction validation rather than trusted intermediaries. Once initialized, parameters are immutable, state transitions are linear, and all contributions, distributions, and loan decisions are verifiable from the ledger.

Execution costs and validation behavior are deterministic under the eUTxO model. No privileged keys exist to move pooled funds, and any deviation from the protocol rules is either rejected by validation or rendered economically disadvantageous.

3 Model

A group is represented by a single on-chain state object that is consumed and recreated on each valid action. Participants authenticate actions using membership tokens while retaining self-custody of assets outside the contract address, and collateral is isolated per participant to reduce shared-state contention. The protocol supports ROSCA mode, where fixed contributions fund periodic distributions, and ASCA mode, where contributions accumulate into a lending pool and loans are decided by voting.

4 State Machine

The protocol is defined as a finite state machine.

4.1 States

The group state progresses through `INIT` (created, parameters fixed), `OPEN` (members can join), `ACTIVE` (contributions, distributions, and ASCA loan actions), `ENDED` (group terminated), and `CLOSED` (final withdrawals complete).

4.2 Transitions

Only predefined actions may advance the state. Invalid transitions are rejected by validation rules. This guarantees that history cannot fork and rules cannot be altered mid-execution.

Table 1: State–Action Matrix (Condensed)

State	Valid Actions
INIT	Join
OPEN	Open, Start
ACTIVE	*Common:* Contribute, Leave, Kick
	ROSCA: Distribute
	ASCA: ApplyLoan, Vote, Borrow, Repay, Liquidate
ENDED	Withdraw
CLOSED	— (terminal state)

Figure 1 illustrates the canonical lifecycle of a group. State transitions are linear and irreversible: each transition consumes the current state UTxO and produces its unique successor carrying the same State Thread Token. Recurring operations occur only in the `ACTIVE` state and do not alter the global lifecycle.

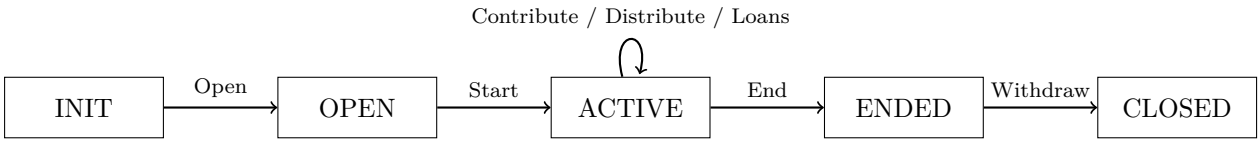

Figure 1: Canonical state progression (recurring operations occur in `ACTIVE`).

5 Tokens and Identity

The protocol separates *state*, *authorization*, and *value*. Membership tokens may move freely between participants, while all coordination rules remain enforced by the ledger.

Identity-linked metadata follows the reference-token construction defined in CIP-68, a transferable user token is associated with a non-transferable reference token locked in a script-controlled UTxO.

5.1 State Thread Token

Each group is identified by a unique non-fungible State Thread Token (STT) held in exactly one script-controlled state UTxO at any time. Any valid transition spends that UTxO and recreates a successor state UTxO carrying the same token, enforcing a single canonical history and preventing state forks.

5.2 Membership Tokens

Participants are authorized to act by holding a membership token. Possession of this token grants the right to perform protocol actions such as contributing, voting, or withdrawing.

Each membership token is paired with a CIP-68 reference token carrying identity and status metadata. The reference token is locked in a script-controlled output, while the membership token remains transferable in the participant's wallet.

This separation makes authorization explicit and non-forgeable while allowing membership status to evolve without reminting tokens.

Membership tokens do not represent value. They represent permission, enforced solely by validation rules.

By binding the membership to tokens instead of public keys, it opens the possibility of script type membership. In other words, a participant may be a script.

5.3 Membership Minting

Membership follows CIP-68 and is minted under `Action.Init` for the group creator, and `Action.Join` for the group members. The join transaction spends the STT state UTxO, updates `SttDatum` to add the participant, and mints a CIP-68 pair: a transferable membership token and a non-transferable reference token locked alongside the initial collateral. `Action.End` (for the group creator) and `Action.Withdraw` (for each group member) burns both tokens and releases collateral.

5.4 Loan Tokens (ASCA)

In ASCA mode, each loan application is represented by a UTxO holding a distinct **loan token**. The loan token identifies the loan terms, vote state, and is associated with a participant.

By isolating loan state from group state, the protocol permits concurrent loan voting and repayment without shared mutable state as much as possible.

6 Protocol Actions (Mechanism)

All actions are validated by comparing input state to output state and enforcing invariant preservation. Each transition is selected by an `Action.*` redeemer and is valid only if the input `SttDatum` and output `SttDatum` satisfy the action-specific invariants.

6.1 STT Policy (Init/End)

The State Thread Token is minted only at initialization and burned only at termination, enforcing a single canonical state for the lifetime of the group.

Algorithm 1 STT Mint/Burn Policy

Require: Exactly one STT minted (+1) at Init or burned (-1) at End.

if Init **then**
    Require consumption of a one-time nonce UTxO input.
    Require parameter sanity checks.
else if End **then**
    Require spending the current STT state UTxO.
    Require deactivation of group memberships.
else
    Reject.
end if
return valid.

6.2 Join and Start

During Join action, a member provides the initial collateral and receives a membership token (CIP-68 User Token). During Start, the group transitions to ACTIVE and sets the time parameter (for ROSCA: distribution schedule; for ASCA: unlock date / lending phase begins).

6.3 Contribute

A contribution is an authenticated payment to the group state UTxO (or designated pool UTxO), updating contribution flags (ROSCA) or counters (ASCA). Multiple contributions may be batched as a single transaction.

7 ROSCA Protocol

In ROSCA mode, participants contribute a fixed amount per round. At distribution time, the pot is allocated to one or more winners depending on the group configurations upon initialization.

7.1 Slashing

Participants who misses contributions are penalized by slashing their collateral up to one contribution amount. If there's not enough collateral, then the participant is marked as ineligible to be nominated as a future potential winner until they rejoin. When a participant rejoins the group, they must deposit the minimum collateral, similar to joining the group.

7.2 Deterministic Winner Selection

The winner selection is pseudorandom and publicly verifiable prior to the distribution, but it is difficult to influence as the entropy is derived from transaction hash and several other group states; selection is performed by repeatedly hashing and modulus to get winning indices from the eligible participants list.

Algorithm 2 Winner Selection (multi-winner, condensed)

Require: Eligible participants list L
Require: Number of winners k
Require: Dynamic entropy E (Tx Hash)
Require: Fixed entropy F (some group states)
1: $W \leftarrow []$
2: **while** $k > 0$ and $|L| > 0$ **do**
3: $h \leftarrow \text{Hash}(E \parallel F)$
4: $i \leftarrow \text{Int}(h) \bmod |L|$
5: append $L[i]$ to W; remove $L[i]$ from L
6: $k \leftarrow k - 1$; $E \leftarrow \text{Hash}(h \parallel \mathit{Serialize}(L))$
7: **end while**
8: **return** W

7.3 Randomness Security Model

This construction is deterministic and auditable. Stronger randomness (e.g., multi-party commit–reveal) may be added in the future.

8 ASCA Protocol

In ASCA mode, contributions accumulate into a pool that funds peer-governed loans.

8.1 Lock Phase and Lending Phase

A lock phase prevents immediate extraction of value by setting an unlock date and requiring a minimum contribution count before a group participant can apply for a loan. Let c_i be participant i's contribution count; eligibility requires:

$$c_i \geq \text{lockPeriod}. \tag{1}$$

8.2 Loan Application and Voting

A loan application creates a dedicated loan UTxO. Each group participant (except for the applicant) may vote once, with voting power proportional to the contribution count:

$$\text{VotingPower}_i = c_i. \tag{2}$$

A loan is approved if:

$$\sum \text{YesVotes} > \sum \text{NoVotes}. \tag{3}$$

8.3 Borrow, Repay, Liquidate

Borrow action disburses funds if the loan application is approved after the voting deadline passes. Repay action returns the loan principal plus interest (or late interest, if it is a late repayment) to

the pool. If loan repayment is missed, lenders may liquidate collateral pro-rata under divisibility constraints enforced during loan application submission.

8.4 Staking Yield

An ASCA group is delegated to a staking pool, configured during group initialization. The stake rewards withdrawals are performed during certain actions enforced through validator constraints. It is withdrawn to the group's pool of funds, adding to the available amount a participant may borrow. It is not directly distributed to each group participant.

9 Incentives and Security

The protocol replaces social enforcement with economic and cryptographic enforcement.

9.1 Incentives

Protocol rules are enforced economically through validation and collateral, not discretion. In ROSCA mode, missed contributions trigger deterministic collateral slashing and loss of eligibility. In ASCA mode, loan obligations are enforced by time-bounded repayment and pro-rata collateral liquidation.

A fixed entry fee when joining a group may be specified at initialization to compensate group creation costs. Voting power is proportional to the contribution count, and deviations from protocol rules are either rejected by validation or non-beneficial to the actors.

9.2 Security Properties

The protocol has no privileged keys that can move pooled funds outside the script rules. Linear state progression is enforced by the STT singleton, and validation behavior is deterministic under the eUTxO model.

Table 2: Threats and Mitigations (condensed)

Threat	**Vector**	**Mitigation**
Platform theft	Custodial control	No custody; funds locked to a unique contract address associated to each participant
State forking	Competing histories	STT singleton; UTxO forgery prevention
Sybil voting	Fake identities	Voting power proportional to contribution count
Griefing	Spam tx attempts	Transaction fees; delays for certain actions enforced by the validators
Distribution bias	Executor influence	The deterministic selection is difficult to influence; extendable randomness

10 Implementation Notes (Minimal)

The protocol is implementable as a set of validators and minting policies covering STT mint/burn, membership minting and reference spending, group state transitions for ROSCA and ASCA actions, and, in ASCA mode, loan application and loan-state validation. Transaction fees are deterministic under the eUTxO model and scales with transaction size and execution units.

11 Conclusion

We described a protocol implementing cooperative savings and credit associations as deterministic state machines enforced by the ledger. The design removes the need for trusted organizers while preserving ROSCA and ASCA mechanics. Linear state progression, explicit collateralization, and contribution-weighted governance provide verifiable execution under predictable validation costs.

Related Work

Rotating and accumulating savings associations have been studied extensively in economics and development literature, where they are shown to provide commitment and liquidity in the absence of formal credit markets. Prior work analyzes their incentive structure, risk-sharing properties, and limitations under social enforcement.

Blockchain-based implementations of cooperative finance typically rely on custodial contracts or off-chain coordination. Akyba differs by encoding ROSCA and ASCA rules as a deterministic on-chain state machine under the eUTxO model.

These constructions are well documented in the literature listed below.

References

[1] S. Ardener, "The Comparative Study of Rotating Credit Associations," *Journal of the Royal Anthropological Institute*, vol. 94, no. 2, pp. 201–229, 1964.

[2] T. Besley, S. Coate, and G. Loury, "The Economics of Rotating Savings and Credit Associations," *The American Economic Review*, vol. 83, no. 4, pp. 792–810, 1993.

[3] F. J. A. Bouman, "Rotating and Accumulating Savings and Credit Associations: A Development Perspective," *World Development*, vol. 23, no. 3, pp. 371–384, 1995.

[4] M. M. T. Chakravarty et al., "The Extended UTXO Model," WTSC, 2020. `https://iohk.io/en/research/library/papers/the-extended-utxo-model/`

[5] Cardano Foundation, "CIP-68: On-chain token metadata via reference tokens," 2022. `https://cips.cardano.org/cips/cip68/`

[6] TxPipe, "Aiken: A Modern Smart Contract Language for Cardano," 2023. `https://aiken-lang.org/`