# Contractless: A Fair Proof-of-Work Multi-Asset Blockchain

Contractless Community
www.contractless.community

**Abstract.** Contractless is a peer-to-peer fair-proof-of-work blockchain consensus that was created to provide equality in mining participation. It supports native multi-asset transaction types and deterministic chain correction without relying on smart contracts. Blocks are distributed through torrent metadata; validated by timestamp, nonce and verifiable random function ordering; and synchronized through a bounded orphan correction process. The system limits nonce searching per round to 256 attempts, reducing specialized hardware advantage, uses miner participation rules to slow early reward concentration, and supports transfers, token issuance, non-fungible tokens, swaps, collateral loans, burns, marketing transactions and vanity address registration as native protocol operations.

# 1 Introduction

Most blockchain systems provide extensibility through general-purpose smart contracts or external application layers. While flexible, this approach has frequently introduced security risks when writing contracts and makes common operations dependent upon execution environments and application-specific validation rules. Contractless takes a different approach by making common blockchain utilities native transactional protocol operations. Transfers, token issuance, non-fungible tokens, swaps, collateralized loans, burns, marketing transactions and vanity address registration are represented directly as transaction types and validated by the base protocol.

Proof-of-work remains a simple and durable mechanism for ordering blocks, but unrestricted nonce search can favor specialized hardware and concentrate rewards among miners with greater computational resources. Contractless uses a new proofing algorithm called Fair-Proof-of-Work which limits each mining round to a one-byte nonce search and uses additional fairness rules to reduce early reward concentration. New miners must first contribute 100 blocks without receiving the base reward and transactions are excluded during mining so block creation remains focused on consensus rather than transaction selection.

Nodes may temporarily disagree about the best block at a given height, especially when competing blocks are discovered close together. Contractless resolves these disagreements through deterministic orphan correction. Competing blocks are compared by timestamp, nonce and verifiable random function output, allowing nodes to select the same preferred block without relying on subjective peer preference or longest-chain replacement.

Blocks are propagated using torrent metadata rather than large inline block messages. This allows peers to announce and request block data while keeping the RPC stream compact. Staged torrents give nodes enough local evidence to correct short forks and replay candidate blocks after rollback. The result is a peer-to-peer blockchain intended to provide native utility, fairer proof-of-work participation, and deterministic synchronization while remaining community operated.

# 2 Transactions

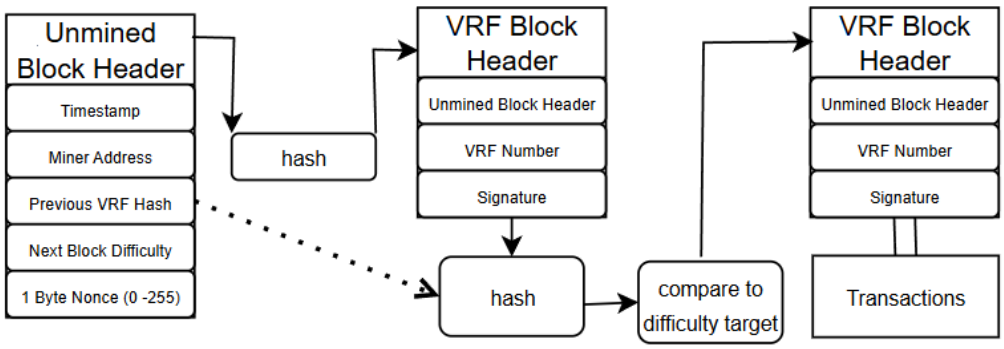
Contractless does not use a general-purpose smart contract layer for common asset operations. Instead, each supported operation is represented as a native transaction type with deterministic validation rules. This keeps transaction behavior inside the base protocol and avoids requiring nodes to execute arbitrary contract code.

Transactions are grouped by purpose. Base transactions transfer value, burn supply and record miner rewards. Asset transactions create tokens, issue tokens and define non-fungible token records. Exchange transactions perform protocol-level swaps. Credit transactions create collateralized loans, record loan payments and manage collateral. Additional native transactions support marketing payments, vanity address registration and genesis initialization.

Because each transaction type is part of the protocol, nodes validate the same operation in the same way. Common blockchain utilities are available without deployed contracts, while invalid transactions can still be rejected before they alter balances, ownership records or loan state.

# 3 Fair Proof of Work

Contractless uses Fair-Proof-of-Work to separate proof construction from transaction selection. A miner builds an unmined block header from the block timestamp, miner address, previous VRF block hash, target difficulty and one-byte nonce. The previous hash references the prior VRF block header, tying each proof to the previous consensus header instead of the previous transaction payload.




For each mining round, workers search the 256 nonce values for a single timestamp. The unmined header is hashed; the miner signs that hash; and the signature is used as both the proof and the input for deriving a verifiable random function number.

The unmined header, VRF number and signature form the VRF block. The VRF block is hashed and the reduced hash value is compared against the current target difficulty. A candidate is valid only when the VRF block hash is below the target. Transactions are assembled only after a valid proof exists.

# 4 Difficulty

Contractless represents difficulty as a target threshold. A VRF block is valid only when its reduced hash value is below the current target. Lower targets are harder to satisfy, while higher targets are easier to satisfy.

Difficulty is adjusted from recent block timing. Let $T_i$ be the timestamp of block $i$, and let $D_i$ be the target carried by block $i$. The rolling average block time is calculated from the time between adjacent blocks:

$$\bar{T} = \frac{1}{n-1} \sum_{i=2}^{n} (T_i - T_{i-1})$$

The rolling average target is the mean of the sampled block targets:

$$\bar{D} = \frac{1}{n} \sum_{i=1}^{n} D_i$$

Contractless uses a target window of 14 to 16 seconds. When $\bar{T}$ already falls inside this window, the next target is set to $\bar{D}$. Otherwise, the current target changes by at most 30 percent:

$$D_{\text{next}} = \begin{cases} \bar{D}, & 14 \le \bar{T} \le 16 \\ D_{\text{current}} + 0.30 D_{\text{current}}, & \bar{T} > 16 \\ D_{\text{current}} - 0.30 D_{\text{current}}, & \bar{T} < 14 \end{cases}$$

When blocks are found too slowly, the target increases and proof becomes easier. When blocks are found too quickly, the target decreases and proof becomes harder. The bounded adjustment avoids sudden target swings while still pushing the network back toward the desired block interval.

# 5 Mining Fairness

Mining fairness is enforced by limiting both the proof search space and the reward advantage of early miners. Each mining round is bound to one timestamp and one byte of nonce space, so a miner may test only 256 nonce values before waiting for the next timestamp or a chain change. Worker threads split this fixed nonce range, but they do not increase the total number of nonce values available to the miner during that round.

New miners must also prove participation before receiving the base block reward. The first 100 blocks mined by a registered miner create a rewards transaction with zero value. After the miner has reached this participation threshold, later eligible blocks use the normal reward schedule.

Because these zero-value reward blocks are valid blocks, the exact number of coins created by Fair-Proof-of-Work cannot be known before the chain is mined. Total issuance depends on how many miners join, how often new miners pass through the participation period and how often fairness rules cause miners to wait. The emission schedule therefore defines the maximum reward available at each height, while actual issued supply is determined by network participation.

Contractless also limits consecutive block production by the same miner. For a block height $h$, the active fairness interval is

$$i = \left\lfloor \frac{h}{960000} \right\rfloor + 1$$

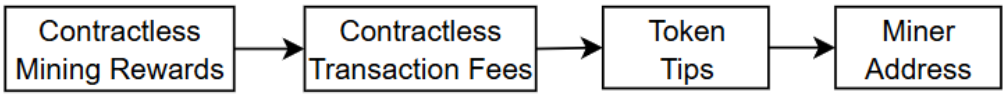
and the maximum allowed consecutive blocks is

$$c = \max(10 - i, 1).$$

Before mining the next block, a node walks backward through the recent headers and counts the current miner's consecutive streak. If the streak is greater than $c$, the miner becomes idle and waits for another miner to advance the chain. This rule starts by allowing up to ten consecutive blocks and tightens over time until the minimum allowance of one consecutive block remains.

# 6 Incentive

The miner incentive is made from three sources: the base mining reward, protocol transaction fees and asset-denominated tips. The reward transaction is created by consensus and is the first transaction in each mined block. When the miner has passed the participation threshold, the reward follows the halving schedule; before that threshold, the reward transaction exists but pays zero.




Transaction fees are paid in the base coin and are credited to the miner when the block is applied to the balance sheet. Each transaction type has its own minimum fee. Base-coin transfers require a minimum fee of 1 percent of the transferred value, while transfers of tokens or non-fungible tokens require a minimum fee of 1 CLC. Other native operations use fixed minimum fees defined by their transaction type.

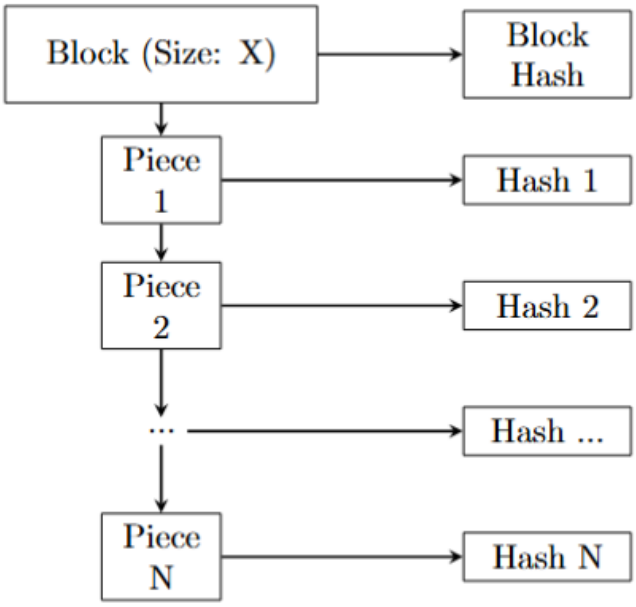
Fees are minimums, not fixed ceilings. A sender may increase the base-coin fee to make the transaction more attractive to miners. Contractless does not require a global "largest fee first" ordering rule; miners still receive the fees paid by the transactions they include, while transaction validity is determined by the protocol's per-type minimums and balance checks.

Tips are used when the miner should receive an incentive in the asset involved in the transaction rather than only in the base coin. Swap transactions carry asset-denominated tips for each side of the trade. Loan payment transactions carry a tip in the loaned asset, and that tip must be at least 1 percent of the payment amount. This allows the miner to be paid from the same asset flow being exchanged or repaid.

All incentive changes are applied through the same balance-sheet process as other transaction effects. If an accepted block is later removed during orphan correction, its reward, fee and tip effects are undone before replacement blocks are replayed.

# 7 Torrent Synchronization

Contractless distributes blocks through compact torrent metadata rather than sending full blocks inline through the RPC stream. A torrent describes one block height and contains the total block length, target difficulty, block timestamp, nonce, VRF number, VRF block hash, piece length, full-block hash, piece hashes and miner address. This lets a node examine the announced block before requesting the full block data.




The full-block hash commits to the entire serialized block. This is the integrity check Contractless uses in place of a Merkle tree: if any transaction, reward, header field or byte of block data is changed, the completed block no longer hashes to the value advertised by the torrent. The VRF block hash identifies the proof header used for mining, while the full-block hash protects the complete saved block.

When a torrent is received, it is staged first so that near-tip competing candidates are not lost during synchronization or orphan correction. The node then validates the metadata that can be checked before downloading pieces.

The piece count must match the advertised block length and piece length; the miner address must be registered for the active network; and the torrent difficulty must match the expected difficulty for that height.

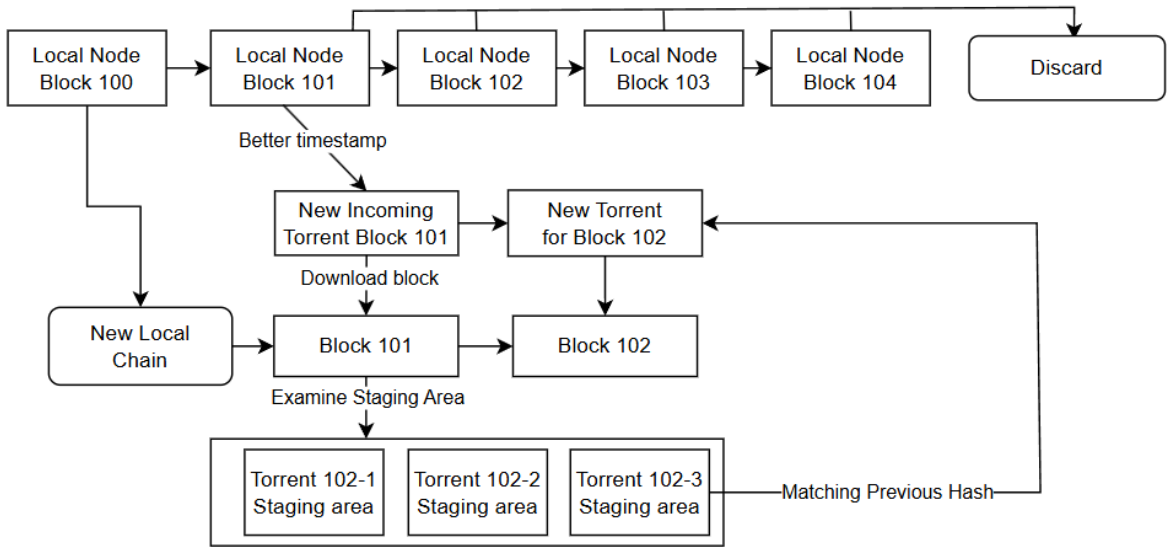
Only after this metadata passes validation does the node request block pieces from connected peers. Each piece is checked against its advertised piece hash before it is accepted into temporary storage. When every piece has been collected, the node combines the pieces and hashes the completed block bytes. The completed block must match the advertised full-block hash before it is decoded, fully verified and saved.

This two-stage process allows bad torrent announcements and bad piece responses to be rejected before they alter chain state. It also allows nodes to keep staged torrent candidates for later orphan correction without treating every received torrent as an immediately accepted block.

# 8 Orphan Correction

Contractless does not use longest-chain replacement as the primary rule for near-tip disagreements. Nodes are allowed to fight over recent blocks inside a bounded orphan window. When two peers differ by ten blocks or fewer, the node examines the shared window of recent block torrents and staged candidates instead of immediately accepting the peer with the larger height.

The comparison is deterministic. For each height in the orphan window, the node compares the local accepted torrent against staged competing torrents. The window is checked from oldest height to newest height so the first block where the local chain loses becomes the rollback point. A competing block wins by earliest timestamp; if timestamps are equal, the lower nonce wins; if nonce is also equal, the lower verifiable random function value wins.



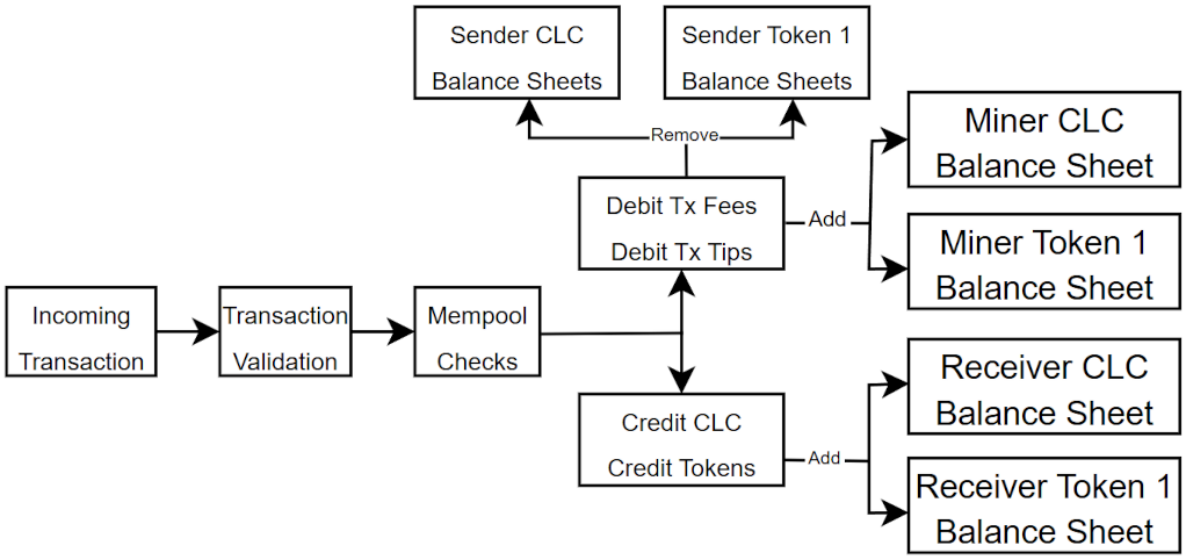

The torrent staging area is what makes this possible. Incoming torrents are saved before they are fully downloaded or accepted, so a node may hold several candidates for the same height. A node does not need to remember which peer supplied a complete path; it only needs the candidate torrents and the deterministic ordering rule.

If a staged candidate beats the local block at the oldest losing height, mining is paused and the chain rolls back to that block's parent. Removed blocks are undone in reverse order, including balance-sheet effects, transaction records, mined-count changes, rewards, fees and tips. Remembered torrent outcomes are then reset because candidates that failed against the old parent may become valid against the new parent.

Replacement proceeds one height at a time. For the next expected height, the node orders staged candidates by status and block-fight fields, then tries the best candidate through the normal torrent download, block verification and save pipeline. If the candidate fails, it is marked invalid and the next best candidate is tried; if it succeeds, other candidates for that height are marked invalid and replay continues.

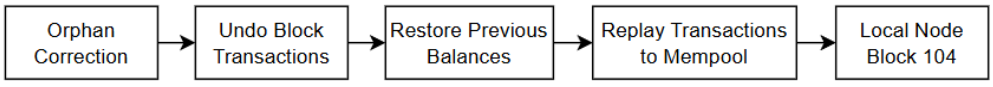
# 9 Balance Sheet

Contractless maintains balances as a derived ledger state rather than as values stored inside account objects. A transaction is first checked against the current balance sheet and protocol records. If it is valid, it may enter the mempool; if it is later included in an accepted block, its effects are applied to the balance sheet during block saving.




This separates transaction validity from mining. Mining proves the block header and determines when a block candidate may be assembled, but the balance changes come from the validated transactions inside the saved block. Transfers debit the sender and credit the receiver; asset operations update token, non-fungible token, loan or collateral records; and miner rewards, fees and tips are applied as transaction effects.

Because the balance sheet is updated from block effects, orphan correction must undo those effects before replaying replacement blocks. When blocks are rolled back, their transactions are reversed in the opposite order they were applied. This restores balances and protocol records to the parent state before the winning staged candidates are downloaded, verified and saved.




The same validation rules are used during live block acceptance and replay. A replacement block cannot alter balances merely because it came from orphan correction; it must still pass the normal block and transaction checks before its effects become part of the active balance sheet.

# 10 Network

Nodes communicate through a binary peer-to-peer RPC stream and use torrent metadata for block distribution. The network operates as follows:

1. New nodes connect to known peers and complete a wallet-signed handshake.
2. Peers announce their reachable node address, miner address, RPC port and current chain height.
3. Nodes reject private, loopback, malformed or otherwise invalid public address announcements.
4. Transactions are submitted through RPC, validated against local protocol state and admitted to the mempool only if valid.
5. Newly discovered blocks are announced as torrent metadata rather than sent inline across the RPC stream.
6. Peers stage torrent candidates, request block pieces, verify piece hashes and verify the completed full-block hash before saving.
7. If competing torrents appear inside the orphan window, nodes run deterministic orphan correction and replay the winning staged path.

A fully participating node must be reachable by other peers, keep its RPC port accessible and maintain reasonably synchronized time. Nodes may continue operating with fewer connections, but poor reachability limits their ability to serve block pieces, receive competing torrent announcements and participate in near-tip correction. Invalid requests, bad torrent data and invalid address announcements may be scored against the sending peer.

# 11 Conclusion

Contractless presents a peer-to-peer blockchain that moves common utility operations into the base protocol instead of relying on general-purpose smart contracts. Native transaction types define transfers, assets, swaps, loans, collateral, burns, marketing payments and vanity address registration with deterministic validation rules shared by every node.

Fair-Proof-of-Work separates proof construction from transaction selection by limiting each mining round to one timestamp and 256 nonce attempts. Miner participation rules, reward delay and consecutive-block limits reduce early reward concentration while preserving proof-of-work block discovery.

Blocks are distributed through torrent metadata, staged before acceptance and replayed through deterministic orphan correction when near-tip disagreements occur. Because balance-sheet effects are applied only through validated block saving and reversed during rollback, replacement paths must pass the same transaction and block checks as live blocks. These mechanisms provide a fair, native multi-asset blockchain intended for community operation without contract execution or centralized block coordination.

# References


[1] Internet Engineering Task Force, R. Braden, "Requirements for Internet Hosts – Communication Layers," *RFC 1122, Section 4.2.3.6, page 101, TCP Keep-Alives*, 1989, https://datatracker.ietf.org/doc/html/rfc1122#page-101

[2] Bram Cohen, "Incentives Build Robustness in BitTorrent," 2003, https://www.bittorrent.org/bittorrentecon.pdf

[3] Satoshi Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2009, https://bitcoin.org/bitcoin.pdf

[4] Bruce Schneier, Stefan Lucks, Niels Ferguson, Doug Whiting, Mihir Bellare, Tadayoshi Kohno, Jon Callas and Jesse Walker, "The Skein Hash Function Family," *Journal of Engineering Research and Applications, Vol. 3, Issue 6, Nov–Dec 2013, pp. 490–495*, https://www.ijera.com/papers/Vol3_issue6/CF36490495.pdf

[5] Sasha Ivanov, "Crypto-platform for asset / custom token issuance, transfer and trading on blockchain," 2016, https://bravenewcoin.com/wp-content/uploads/2023/11/WAVES.pdf