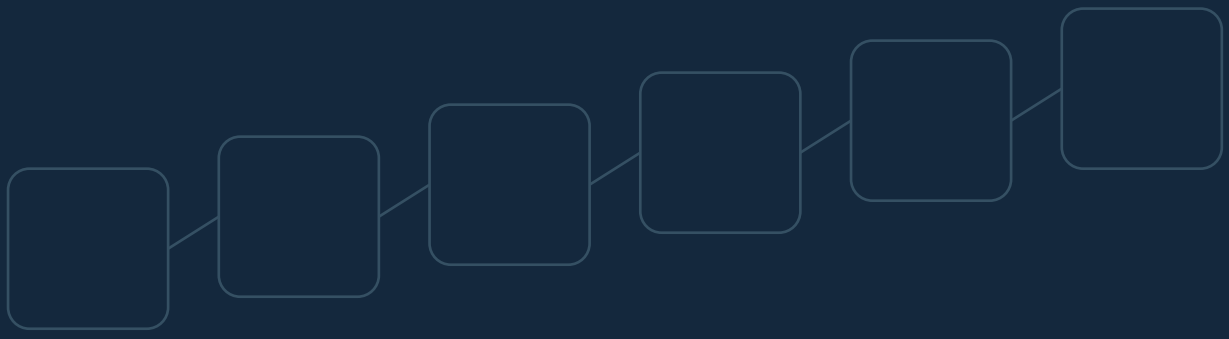
TORVYON / Byte



Torvyon Byte

A Peer-to-Peer Electronic Value System

Aram Jaff
a.j.s-engineer@outlook.com

VERSION	DATE	STATUS
1.0	Mar 3, 2023	Implementation-aligned technical whitepaper

Abstract

A purely peer-to-peer value system would allow online payments to be sent directly from one party to another without requiring a financial intermediary to approve, reverse, or reorder them. Digital signatures provide part of the solution, but the main problem is preventing double-spending without a trusted operator. Torvyon Byte addresses this through a proof-of-work network that timestamps transactions into a public chain of hashed blocks. Transactions transfer spendable outputs by referencing prior outputs, proving authorization with digital signatures, and creating new outputs for the next owners. The network accepts the valid chain with the greatest cumulative proof-of-work. As long as honest miners control more hashing power than cooperating attackers, they will extend the valid chain faster than an attacker can replace it.

1. Introduction

Commerce over communications networks has come to rely heavily on trusted third parties to process electronic payments. While this model is operationally convenient, it leaves final settlement dependent on institutional discretion, mediation costs, and reversible transaction flows. What is needed is a system based on cryptographic proof instead of trust, allowing two willing parties to transfer value directly without requiring a central operator to decide which payment is final.

Torvyon Byte provides such a system through public transaction broadcast, digital signatures, proof-of-work ordering, and independent verification by every node. Rather than asking a central authority to confirm which transfer arrived first, the network records transactions in a public chain of proof-of-work blocks, allowing participants to converge on a single accepted history.

Torvyon exists to prove that a modern chain does not need to weaken consensus to feel fast. Instead of replacing proven consensus with experimental shortcuts, Torvyon keeps the battle-tested foundation of proof-of-work, pairs it with predictable issuance, and uses faster blocks with a native UTXO ledger for simple, transparent settlement.

The design goal is deliberately narrow: a native coin, its own chain, public verification, local signing, and a wallet model that does not require users to trust a payment operator with custody of their keys.

2. Transactions

Torvyon Byte defines an electronic coin as an unspent transaction output protected by a locking script. Each owner transfers value to the next by creating a transaction that consumes one or more prior outputs and creates new outputs for the next owner or owners.

Each transaction currently includes:

- a transaction version
- a chain identifier
- one or more inputs referencing previous outputs
- one or more outputs containing value and locking scripts
- a lock time
- the sender public key and signature for authorization

The current mainnet signing model derives the sender address from the included public key and requires every referenced output to belong to that signer. This binds authorization to the actual key material rather than to a separately claimed sender field.

Figure 1. Transfer of ownership through chained outputs

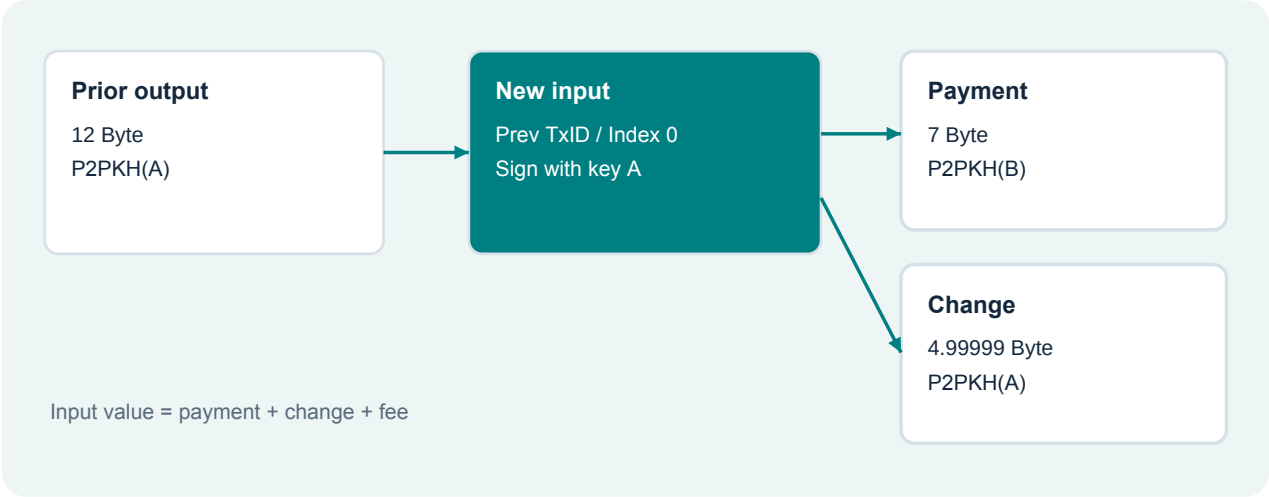


Figure 1 illustrates the transaction model. A transaction spends prior outputs, proves control of them through the signer's key, pays the recipient, and returns any remainder as change to a new output controlled by the sender.

Torvyon mainnet currently uses standard p2pkh outputs for spendable value. The scripting layer also defines multisignature, timelock, and hashlock forms, but the present spend-validation path on mainnet accepts ordinary spendable value through the standard pay-to-public-key-hash form.

Double-spending is prevented because every node checks that each referenced output exists in the current UTXO set and has not already been spent by an earlier accepted transaction.

3. Timestamp Server

The solution to transaction ordering begins with a timestamped chain of blocks. Each block contains a set of transactions and commits to the block before it by including the previous block hash in its header. Each published block therefore serves as a timestamped commitment both to the transactions it contains and to the history before it.

Figure 2. Hashed chain of timestamped blocks

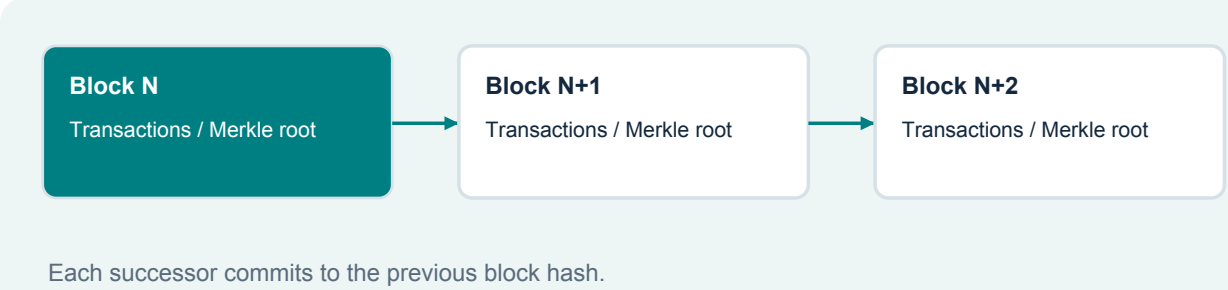
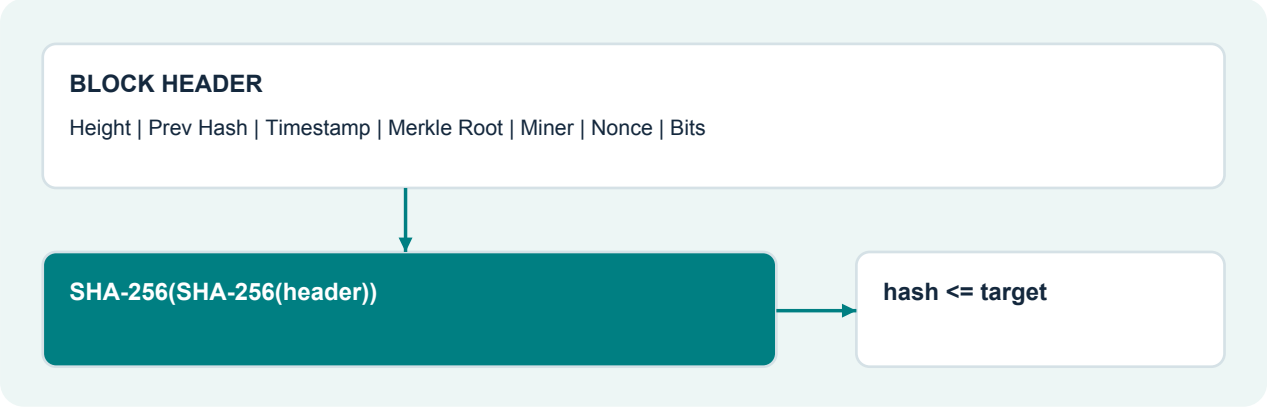


Figure 2 illustrates the timestamp chain. Once a transaction is included in a block, altering it changes the Merkle root, then the block hash, and therefore every later block that depends on it.

4. Proof-of-Work

To implement the timestamp chain on a peer-to-peer basis, Torvyon Byte uses a proof-of-work system. Miners vary a nonce in the block header until the double SHA-256 hash of that header is less than or equal to the target encoded in compact Bits form.

Figure 3. Proof-of-work over the block header



Proof-of-work provides the network with an objective basis for majority decision. Nodes do not count identities, addresses, or administrative votes. They follow the valid chain with the greatest cumulative work.

Torvyon's current mainnet parameters are:

- target block time: 20 seconds
- difficulty retarget interval: 20 blocks
- initial reward: 2.5 Byte
- reward epoch: 1,576,800 blocks
- reward decay per epoch: 3427 / 3500

The retarget window is deliberately short so ordinary CPU miners can keep the chain responsive under changing local hash power.

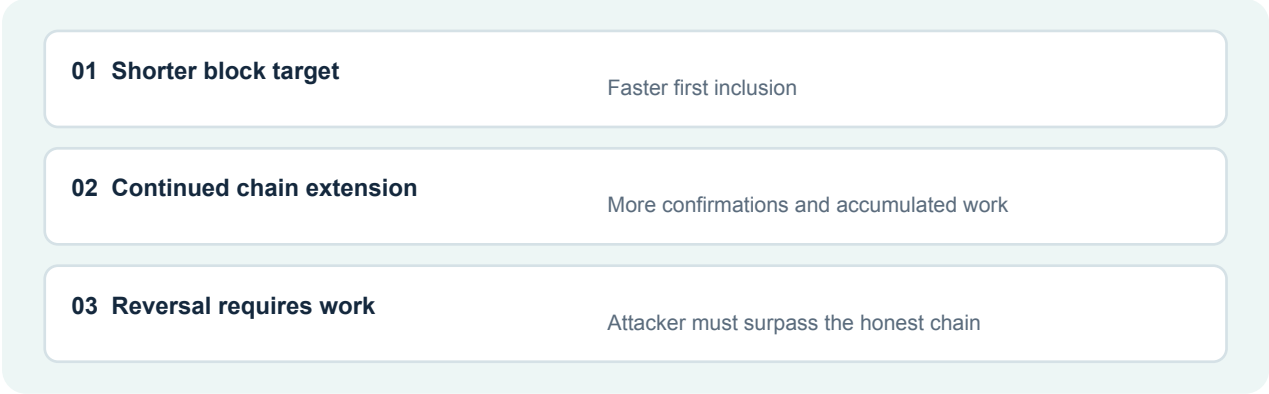
The 20-second block target does not by itself weaken the chain. Security comes from the cost of accumulated proof-of-work needed to replace accepted history, not from making each block individually slow. A shorter block interval changes how confirmations are counted and how often temporary races may occur, but it does not remove the requirement that an attacker must still outwork the honest network to replace the chain.

As long as most of the network's hashing power is controlled by miners who are not cooperating in an attack, they will extend the valid chain faster than an attacker can replace it. As the network grows and honest cumulative work increases, rewriting settled history becomes progressively less practical.

In Torvyon's design, the shorter interval is made practical by four constraints:

- blocks are small enough to propagate quickly across the intended network
- difficulty is retargeted every 20 blocks, so the chain can react quickly to changing hash power
- nodes choose the chain by cumulative work, not raw block count
- users rely on confirmation depth for finality, not on a single block alone

Figure 3A. Why a 20-second block target can still be secure



A payment therefore should not be judged as final merely because it appeared in one fast block. It becomes increasingly secure as additional valid blocks are built on top of it. The 20-second target improves responsiveness, while the security model remains the same: reversing settled history requires redoing and surpassing the proof-of-work of the honest chain.

Figure 3B. Security accumulation across 20-second confirmations

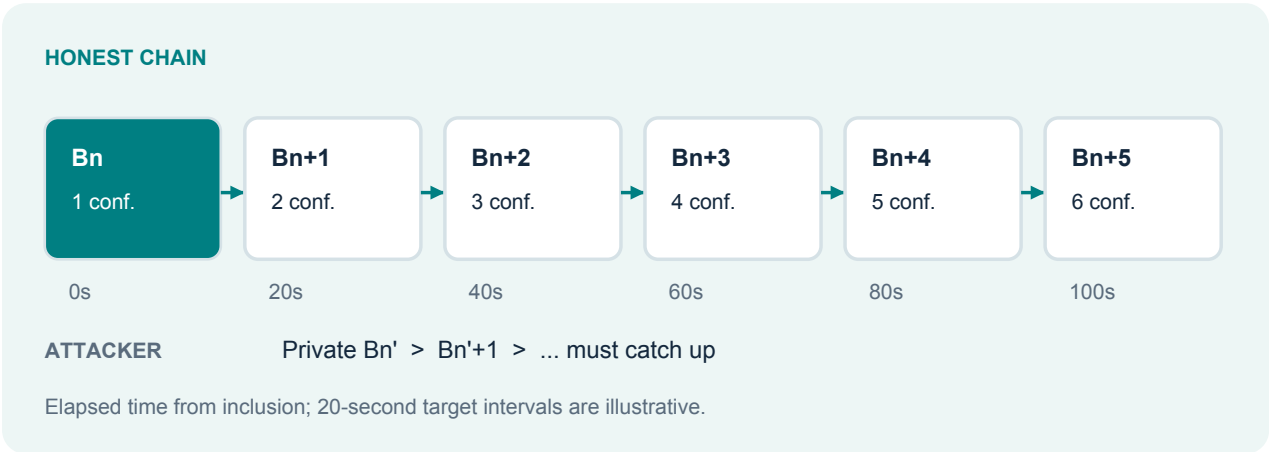


Figure 3B shows the practical meaning of a 20-second target. Finality does not reside in the first block alone. It increases as the transaction becomes buried under additional proof-of-work. A shorter block interval simply allows the network to accumulate those confirmations more frequently.

The main engineering constraint is propagation. If blocks take too large a fraction of the block interval to reach miners, competing tips become more common and honest work fragments across temporary forks. Torvyon therefore keeps the block cadence short only together with fast propagation assumptions and a short retarget window. Under those conditions, the network can gain the user experience of faster inclusion without changing the basic security rule that the heaviest valid chain wins.

5. Network

The network is a peer-to-peer mesh using TCP transport and JSON messages. Transactions and blocks are broadcast on a best-effort basis. The steps are:

1 New transactions are broadcast to peers.

2 Each node verifies signatures and structural correctness.

3 Valid transactions enter the mempool.

4 A miner assembles a candidate block from pending transactions.

5 The miner searches for a nonce satisfying the current target.

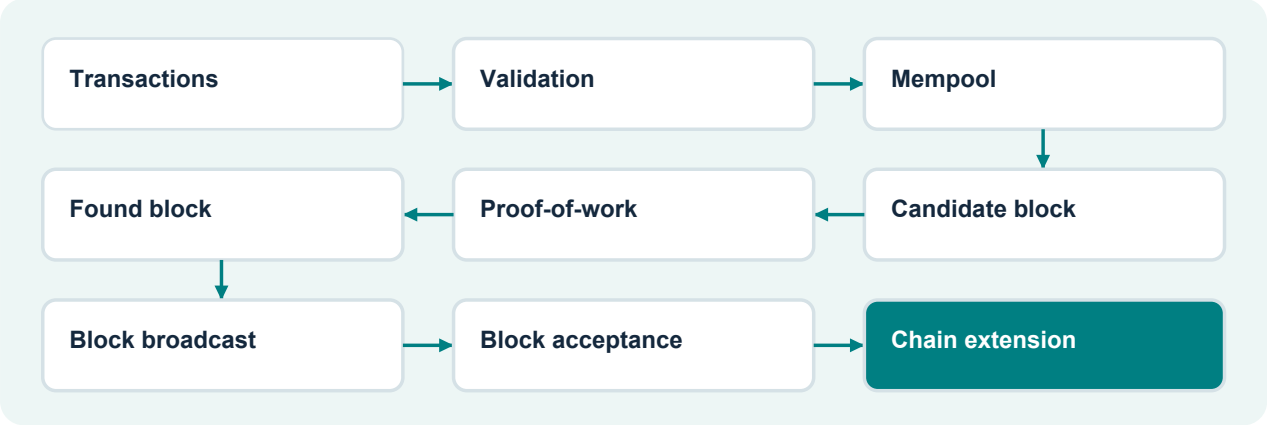
6 The mined block is broadcast to peers.

7 Peers accept the block only if its linkage, proof-of-work, transactions, and reward rules are valid.

8 Nodes continue mining on the accepted chain with the greatest cumulative work.

For operational bootstrapping, Torvyon distinguishes between the protocol's fixed genesis block and any long-lived public bootstrap node. Genesis is a historical consensus object at height 0. A public bootstrap node, sometimes called an anchor node, is simply an always-on public peer that helps new nodes import the already-live chain's genesis and discover other peers. Replacing one public seed with another does not change consensus, chain history, or asset ownership, provided the replacement node joins the same chain rather than starting a new solo network.

Figure 4. Network flow



If two valid blocks are found at nearly the same time, some nodes may temporarily work on different branches. The branch that accumulates more proof-of-work first becomes the accepted chain, and nodes reorganize to it if necessary.

Torvyon nodes also announce a dialable peer address when configured with a public p2p-advertise address. New nodes can use one or more seed peers to join the network, import the live genesis identity, sync historical blocks, and learn additional peer addresses through gossip. A node that is configured with seed peers deliberately waits until it has observed the network before mining, reducing the risk that a fresh install accidentally races ahead on an isolated local fork.

5.1 Network Security Controls

The supplied technical specification describes the following peer-to-peer controls. These implementation descriptions have not been independently verified against source code for this revision.

Peer diversity and connection limits. Nodes track peer counts, unique IP addresses, and subnet distribution using /24 groups for IPv4 and /64 groups for IPv6. Inbound connections are limited per IP and subnet to reduce concentration. Diversity alerts help identify possible isolation, but different addresses or subnets do not prove independent ownership.

Network compatibility checks. The peer handshake checks protocol version, chain ID, genesis hash, and network magic. A verified peer in this context has matching network parameters; it is not necessarily honest or independently operated. An attacker using the same parameters can pass these checks.

Fork and partition monitoring. Nodes compare peer-reported tip hashes and heights and track synchronization timeouts, fork events, and stale blocks. These signals can reveal disagreement or stalled synchronization. Temporary tip differences can also arise during normal propagation, and matching reports do not prove that a node is not isolated.

Operational visibility. The specification describes live security metrics through /api/security and /status, with a corresponding wallet Stats view. Operators should interpret these metrics together with propagation behavior and the availability of independent peers.

Security boundary. These controls help reduce the risk of Sybil, eclipse, and network partition attacks and detect signs of network disruption. Resistance to history rewriting depends on honest miners maintaining sufficient cumulative proof-of-work. Peer diversity alone does not establish mining decentralization or prevent a competing valid chain with greater cumulative work from becoming the accepted history.

6. Incentive

The first spendable value on the chain is created in genesis allocations and in subsequent block subsidies. Torvyon mainnet starts from a fixed genesis allocation of 21,000,000 Byte distributed across Foundation, Ecosystem, and Liquidity wallets, with vesting applied to the long-term treasury addresses.

After genesis, each valid block pays:

- the scheduled block subsidy
- the sum of transaction fees in that block

The current monetary ceiling is fixed by consensus at:

210,000,000 Byte

with:

1 Byte = 100,000,000 nibbles

The chain counts block 0's scheduled subsidy in the total issuance curve but does not credit that block 0 subsidy to a spendable wallet. Real miner payouts begin from block 1 onward.

This creates three distinct sources of spendable value:

- genesis allocation UTXOs created once at chain bootstrap
- miner reward UTXOs created by valid blocks at height 1 and above
- ordinary transfer UTXOs created when one holder spends to another

In UTXO mode, all later spendable value remains in the same form regardless of origin. A wallet funded by genesis, by mining, or by a later transfer still holds spendable outputs rather than a mutable account balance bucket.

Torvyon does not use a stepwise halving schedule. Instead, the block reward decays once per reward epoch by the ratio:

reward(epoch) = floor(initial_reward * 3427^epoch / 3500^epoch)

This decay curve was chosen so that the lifetime subsidy converges exactly to 189,000,000 Byte, which combined with the 21,000,000 Byte genesis allocation matches the 210,000,000 Byte maximum supply.

7. Reclaiming Disk Space

Transactions in each block are summarized by a Merkle root rather than by embedding every transaction directly into the block hash.

Figure 5. Transactions hashed into a Merkle tree

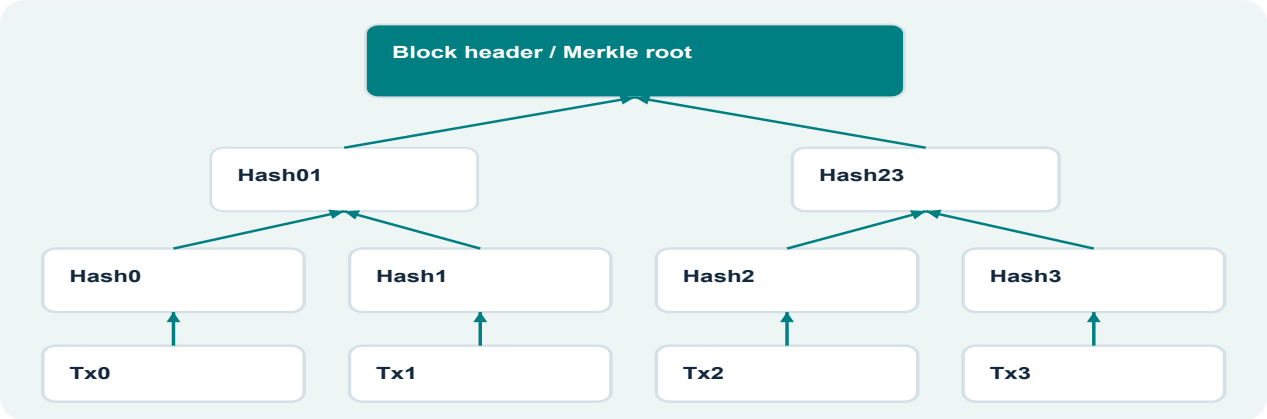


Figure 5 shows how transaction hashes are reduced to a single root committed by the header. This allows historical verification paths to be represented compactly.

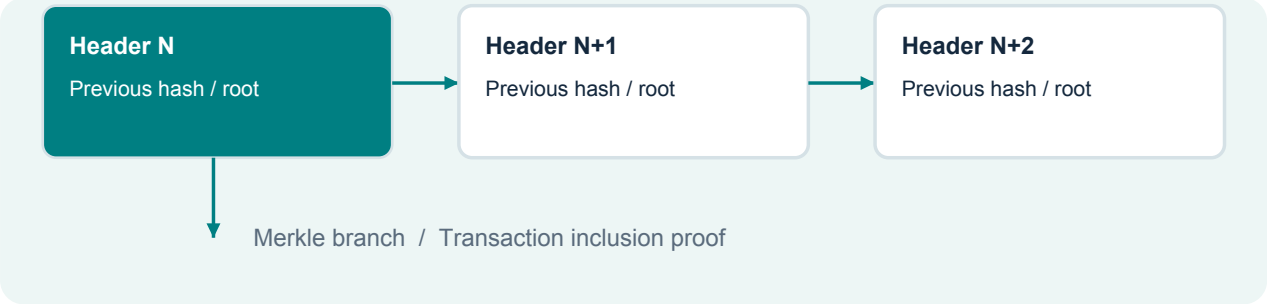
Torvyon's current implementation stores full blocks and a persisted UTXO set. The presence of Merkle commitments allows later pruning strategies and lightweight proofs without changing the block-hash design.

8. Simplified Verification

A lightweight client can verify that a transaction was accepted by checking:

- that the transaction appears in a block
- that the block is part of the heaviest valid chain
- that enough later blocks confirm it

Figure 6. Header chain and transaction proof



The current Torvyon wallet and explorer behavior is still full-node-oriented at the API layer. Confirmation state is exposed directly by the node, and the public interfaces today emphasize full validation and UTXO queries rather than a separate header-only light-client protocol. The chain format, however, already supports Merkle-branch-based proof construction.

9. Wallets and Public APIs

Torvyon's wallet model is not tied to one official application. The PWA wallet, CLI tools, explorers, exchange services, and future third-party wallets all use the same public node interface and transaction rules.

The recommended wallet flow is:

1 Generate or import a secp256k1 private key locally.

2 Derive the x-only Schnorr public key and the corresponding bt1 address.

3 Query spendable outputs for that address.

4 Build a UTXO transaction with payment and change outputs.

5 Sign the transaction locally under the torvyon-mainnet-1 signing domain.

6 Submit only the finished signed transaction to the network.

No normal wallet needs to send a private key to a node. A node only needs the signed transaction, the public key, and the referenced UTXOs in order to verify authorization.

Current public interfaces include:

- /status for chain and node status
- /account/{address} for address-level state
- /api/address-utxos/{address} for spendable outputs
- /api/address-transactions/{address} for wallet history
- POST /tx for signed transaction broadcast
- /tx/{hash} and /api/tx-confirmation/{hash} for transaction lookup
- /api/supply for circulating, total, and maximum supply information
- /api/peers for peer discovery and diagnostics

The user-facing network name should be displayed as Torvyon Mainnet. Internally, the signing domain remains torvyon-mainnet-1 so transactions are cryptographically separated from any future testnet, fork, or mainnet replacement.

9.1 Local Key Storage and Signing

The supplied technical specification describes local key generation and transaction signing in the wallet. Normal payment submission sends a signed transaction rather than the private key. Local signing reduces custody dependence but still requires trustworthy wallet code and a secure device.

Password vault. The described password vault uses PBKDF2-HMAC-SHA-256 with 150,000 iterations and a random salt to derive an AES-256-GCM encryption key. Protection against offline guessing depends on password strength and the derivation cost. This description is not a certification that the current parameters are sufficient for every device or threat model.

Biometric unlock limitation. In the described biometric vault, the AES key is stored alongside the encrypted private key in the same database record. Face ID or Touch ID gates the application unlock flow but is not cryptographically bound to decryption. Access to both stored values can bypass that gate; this design does not provide hardware-backed isolation of the wallet key.

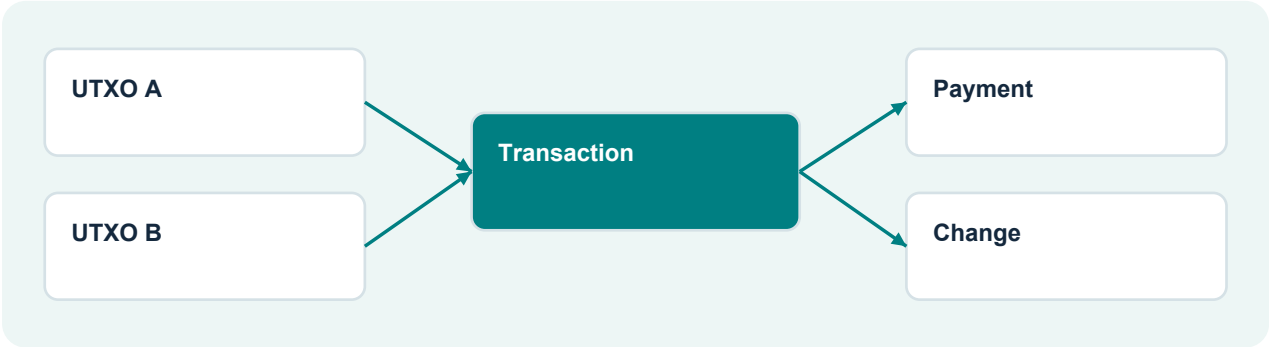
Active-session exposure. The described wallet keeps the decrypted key in sessionStorage and a JavaScript variable during use. Scripts executing in the application origin may access it, so script injection or compromised application code can expose funds. Session lifetime is not a cryptographic protection or a guarantee of secure erasure.

Hardening direction. Reduce decrypted-key lifetime, clear application references on lock, protect against script injection, and use a signing design that keeps secrets outside ordinary browser-accessible storage where practical. These are recommended improvements, not claims that such protections are already implemented.

10. Combining and Splitting Value

Value is combined and split through multiple inputs and multiple outputs.

Figure 7. Combining and splitting outputs



This model avoids maintaining a mutable balance object as the primary consensus unit. Instead, the UTXO set records exactly which coins remain spendable. Wallets may aggregate many smaller outputs into one transaction or split one larger output into payment and change outputs as needed.

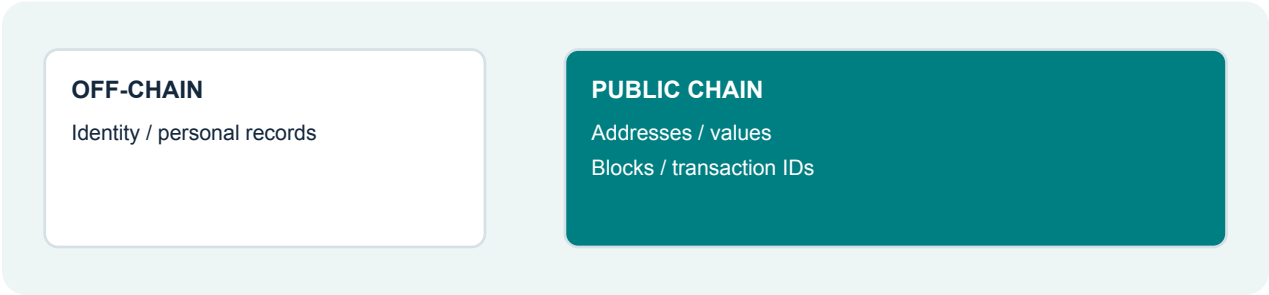
Torvyon's current send flow already constructs change outputs automatically when the selected inputs exceed the payment amount plus fee.

Operationally, Torvyon benefits from wallet role separation even though the consensus rules treat all spendable UTXOs the same. A transparent deployment keeps long-term treasury, ecosystem reserve, market-liquidity inventory, exchange hot-wallet payouts, registrar operations, and mined rewards in separate wallets wherever practical. This is not a protocol requirement; it is an accounting, auditability, and risk-control recommendation.

11. Privacy

The network requires transactions to be public so that every node can validate the same history. Privacy therefore comes not from hiding transaction existence but from limiting explicit identity binding at the protocol layer.

Figure 8. Public transaction model



Torvyon addresses are derived from public keys, and the chain does not require real-world names for ordinary value transfer. At the same time, privacy is not absolute:

- repeated reuse of the same address is linkable
- multi-input spending can reveal common control
- any off-chain identity association can expose a transaction cluster

For that reason, the recommended privacy model remains operational rather than magical: separate addresses, minimal identity leakage, and keeping personal records off-chain. Torvyon's optional data field is size-bounded specifically to avoid turning the chain into a raw document store.

12. Security Model and Limitations

Torvyon's security comes from local key control, deterministic transaction verification, proof-of-work, and cumulative-work chain selection. These properties reduce reliance on trusted intermediaries, but they do not remove all operational risk.

Important limitations:

- one confirmation is fast inclusion, not final settlement
- security increases with confirmation depth and honest cumulative work
- a small public peer set increases partition and isolated-mining risk
- wallet software must protect private keys locally
- the chain cannot recover a lost key or reverse a valid signature
- browser/PWA wallets should minimize how long decrypted keys remain in memory
- public transaction history is permanently visible

For higher-value custody, the recommended direction is a TorvyonKey-style flow: the wallet may display balances and history from public data, but spending requires the user to present the private key or an external encrypted key file at the moment of signing. The key should be used only to produce the signature and then cleared from active memory.

13. Calculations

The security race between the honest chain and an attacker chain can still be modeled as a proof-of-work catch-up problem. Let:

- p be the fraction of total hash power controlled by honest miners
- q be the fraction controlled by an attacker
- z be the number of blocks the attacker is behind

The probability that the attacker ever catches up from z blocks behind is:

```
q_z = 1 if p <= q
q_z = (q / p)^z if p > q
```

If the honest chain stays ahead and p > q, the attacker's success probability falls exponentially as confirmations increase.

For a 20-second block target, confirmation depth can be translated directly into elapsed wall-clock time:

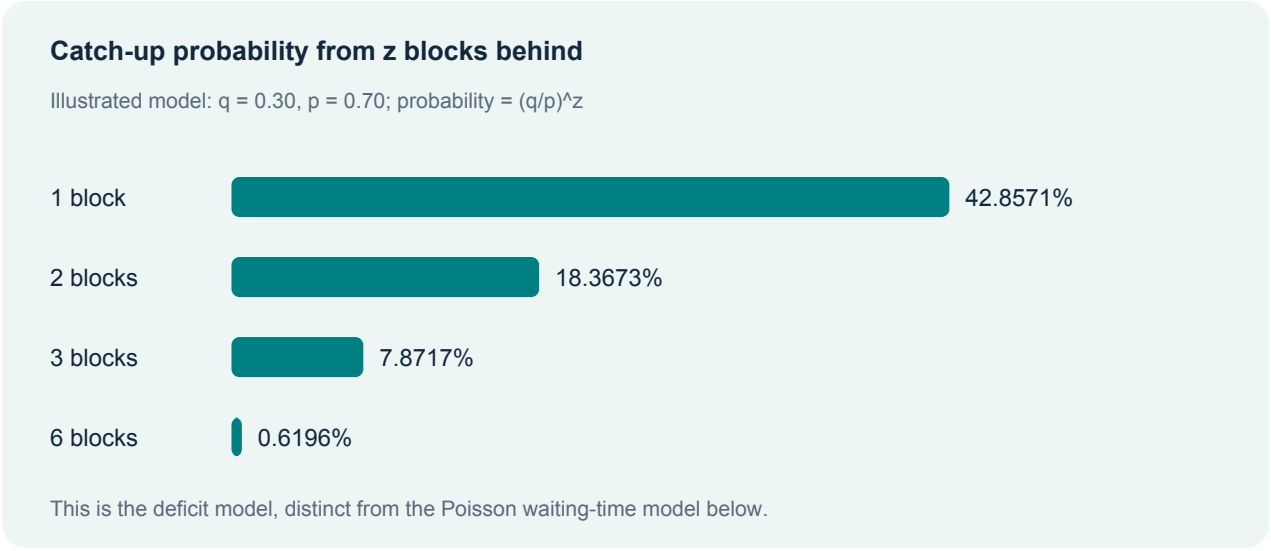
```
z confirmations ~= 20z seconds
```

So if a receiver waits:

- 1 confirmation, the wait is about 20 seconds
- 3 confirmations, the wait is about 60 seconds
- 6 confirmations, the wait is about 120 seconds
- 12 confirmations, the wait is about 240 seconds

The security quantity is still z, not the raw number of seconds. The time only matters because it tells us how quickly the chain can accumulate more blocks of honest work.

Figure 9. Confirmation depth versus attacker success



Example: if an attacker controls q = 0.30 of the hash power and the honest network controls p = 0.70, then:

```
q/p = 0.30 / 0.70 = 0.428571...
z = 1 -> (q/p)^1 = 0.428571
z = 2 -> (q/p)^2 = 0.183673
z = 3 -> (q/p)^3 = 0.078717
z = 6 -> (q/p)^6 = 0.006196
```

At a 20-second target, those confirmation waits correspond roughly to:

- 1 confirmation: 20 seconds
- 2 confirmations: 40 seconds
- 3 confirmations: 60 seconds
- 6 confirmations: 120 seconds

The important property is that the attacker's success probability drops by depth. A shorter block interval means the same confirmation mathematics can be reached sooner in wall-clock time, provided the network can propagate blocks fast enough that honest miners are usually extending the same tip rather than wasting large fractions of work on accidental races.

Using the same Poisson approximation for the attacker's possible progress while the receiver waits for z confirmations:

```
lambda = z * (q / p)
```

and:

$$P_{\text{rev}}(z) = 1 - \sum_{k=0}^{z} \text{Pois}(k;\lambda)\,(1 - r^{z-k})$$

```
where r = q / p  and  Pois(k; lambda) = lambda^k * e^-lambda / k!
```

These formulas do not replace engineering judgment, but they explain the basic confirmation logic of the network: deeper inclusion under honest majority hashing makes reversal rapidly less probable.

Bitcoin's own whitepaper (Section 11) derives this starting from the infinite-tail form before rearranging it into the finite sum shown above:

$$\sum_{k=0}^{\infty} \frac{\lambda^k e^{-\lambda}}{k!} \cdot f(k)$$

```
where f(k) = (q/p)^(z-k) if k <= z, else 1
```

Interpreting the two probability models. The expression (q/p)^z assumes the attacker is already z blocks behind. The Poisson approximation instead includes possible private mining progress while the receiver waits. The two values answer different questions and must not be presented interchangeably as a network security score.

For q = 0.30 and p = 0.70, an attacker already six blocks behind has a catch-up probability of about 0.6196%, with a complementary failure probability of 99.3804%. Under the Poisson waiting model at z = 6, the modeled success probability is about 13.2111%, with a complementary failure probability of 86.7889%.

These are conditional estimates, not measured guarantees for Torvyon. They assume stable hash-power shares and the stated mining model. Actual assessment also requires mining concentration, available competing hash power, propagation and stale-block behavior, network connectivity, and implementation correctness. Confirmation times are averages under the target cadence, not deadlines. No fixed percentage follows from the 20-second target or peer count alone.

Conclusion

Torvyon Byte is a proof-of-work electronic value system built around public transaction broadcast, UTXO-based ownership transfer, Merkle-committed blocks, and longest-chain consensus by cumulative work. Its current implementation uses compact block headers, deterministic transaction hashing, standard P2PKH output spends, fee-backed mining incentives, a finite monetary ceiling, and an offline-generated genesis UTXO allocation for fresh-chain bootstrapping.

The system does not rely on a trusted payment operator to decide finality. Instead, finality emerges from signatures, proof-of-work, public verification, and continued extension of the valid chain with the greatest accumulated work.

Converting to Go code (internal/pow/security.go):

```
func AttackerCatchupProbability(q float64, z int) float64 {
    p := 1.0 - q
    lambda := float64(z) * (q / p)
    sum := 1.0
    poisson := math.Exp(-lambda)
    for k := 0; k <= z; k++ {
        if k > 0 {
            poisson *= lambda / float64(k)
        }
        sum -= poisson * (1 - math.Pow(q/p, float64(z-k)))
    }
    return sum
}
```