

BitcoinBT (BTCBT) — Technical Whitepaper

Base implementation: Bitcoin Core v26

Fork activation height: 903,844

Table of Contents

1. Chain Origin and Consensus Transition Structure
2. Block Validity and Proof-of-Work Model
3. Difficulty Adjustment System (ASERT)
4. Block Resource Limit Model (Size · Weight · SigOps)
5. Monetary Policy and Subsidy Structure
6. Security Model and Chain-Selection Principle
7. Fork Activation Boundary and State Transition
8. Node Validation Determinism and Consensus Convergence
9. UTXO State Transition Model
10. Transaction Validity and Coinbase Rules
11. Network Protocol and Block Propagation
12. Consensus Constants and Parameter Fixity
13. Consensus Invariance and Structural Conclusion

1. Chain Origin and Consensus Transition Model

1.1 Definition of Chain Inheritance

BitcoinBT (BTCBT) is a SHA-256d proof-of-work network that inherits the chain state of Bitcoin exactly up to Bitcoin block height **903,844**.

The chain state at block height h is defined as the following tuple:

$$\text{State}_h = (\text{UTXO}_h, \text{HeaderChain}_h, \text{ScriptRules}_h, \text{DifficultyState}_h)$$

Where:

- UTXO_h : the set of unspent outputs up to height h
- HeaderChain_h : the linked structure of block headers
- ScriptRules_h : the script validation rules
- DifficultyState_h : the difficulty state at that point

BTCBT does **not** create a new genesis block. It inherits the existing Bitcoin chain continuously and forks **only** by switching consensus rules as a function of height.

1.2 Consensus Transition Function

Consensus rules are determined **solely** by block height. The transition is defined by the fixed constant `btcbt_fork_block_height`. No version-bits signaling, miner voting, or runtime signaling is used.

For a given height input, all nodes apply the **same** consensus rules.

1.3 Chain Selection Function

Chain selection follows the **maximum accumulated work** principle.

Define the work of a single block (based on its target) and accumulated work as:

$$\text{Work}(b) \propto \frac{1}{\text{Target}(b)}$$
$$\text{ChainWork}(\text{chain}) = \sum_{b \in \text{chain}} \text{Work}(b)$$

Selection rule:

$$\text{BestChain} = \arg \max_{\text{chain}} \text{ChainWork}(\text{chain})$$

A lower Target implies higher Work. All computations are integer-based; no consensus drift occurs.

2. Block Validity Model

2.1 Proof-of-Work Condition

A block B must satisfy:

$$\text{SHA256d}(\text{Header}(B)) \leq \text{Target}_h$$

Where:

- $\text{Header}(B)$: the 80-byte block header
- Target_h : the difficulty target at height h

This enforces computational scarcity; difficulty is controlled by adjusting the target.

2.2 Block Header Structure

BTCBT does not modify the block header structure.

Field	Size
Version	4 bytes
PreviousBlockHash	32 bytes
MerkleRoot	32 bytes
Time	4 bytes
nBits	4 bytes
Nonce	4 bytes

Total: **80 bytes**. Since the structure is unchanged, the SHA-256d PoW definition is unchanged.

2.3 nBits (Compact) Reconstruction

The compact representation nBits is the compressed form of the target.

Let nBits encode an exponent e and mantissa m . The reconstructed target is:

$$\text{Target} = m \cdot 2^{8(e-3)}$$

Validity constraint:

$$\text{Target} \leq \text{PowLimit}$$

PowLimit is the maximum allowed target. If exceeded, the block is invalid. This prevents difficulty collapse.

3. Difficulty Adjustment Overview

3.1 Target Block Interval

BTCBT target mean block interval:

$$T = 300 \text{ seconds}$$

This defines a 5-minute block schedule. Difficulty is adjusted to converge the realized interval toward T .

3.2 Pre-Fork Model

For the pre-fork range, Bitcoin's legacy difficulty adjustment applies:

- retarget every 2016 blocks
- 4× clamp range

BTCBT does not alter this pre-fork segment.

3.3 Post-Fork Model (ASERT)

After the fork, BTCBT uses **ASERT** (Absolutely Scheduled Exponentially Weighted Target).

Define time error:

$$\Delta t = (t_h - t_0) - T(h - h_0)$$

Conceptual ASERT target:

$$\text{Target}(h) = \text{Target}_0 \cdot 2^{\frac{\Delta t}{\tau}}$$

Where:

- h_0 : anchor height
- t_0 : anchor timestamp
- Target_0 : anchor target
- τ : half-life (172,800 seconds)

Implementation uses fixed-point integer arithmetic; floating-point is not used.

3.4 Interpretation

- If blocks are **slow** → Target increases → difficulty decreases
- If blocks are **fast** → Target decreases → difficulty increases

ASERT adjusts per block continuously and removes 2016-block "jump" retargeting.

4. ASERT Difficulty Adjustment Model

4.1 Post-Fork Warmup Window

Immediately after the fork, the first 6 blocks run at a relaxed difficulty fixed to the minimum:

Applied range:

$$h \in [H_f + 1, H_f + 6]$$

In this range:

$$\text{Target}(h) = \text{PowLimit}$$

PowLimit is the maximum target (minimum difficulty). This is a consensus-fixed rule and not runtime-adjustable, preventing chain stalls under abrupt hash-rate drops.

4.2 Anchor Block Definition

ASERT uses a fixed anchor block:

- h_0 : anchor height
- t_0 : anchor timestamp
- Target_0 : anchor target

All nodes use the same anchor values, ensuring deterministic target computation.

4.3 ASERT Exponential Form

Time error:

$$\Delta t = (t_h - t_0) - T(h - h_0)$$

Target:

$$\text{Target}(h) = \text{Target}_0 \cdot 2^{\frac{\Delta t}{\tau}}$$

Half-life:

$$\tau = 172,800 \text{ seconds}$$

Half-life defines the adjustment speed: an error of τ changes difficulty by a factor of 2.
This is a consensus constant.

5. ASERT Boundary Conditions and Integer Implementation

5.1 Target Upper/Lower Bounds

Consensus requires:

$$1 \leq \text{Target}(h) \leq \text{PowLimit}$$

Effective applied target:

$$\text{Target}_{\text{applied}}(h)$$

$$= \min(\text{PowLimit}, m)$$

$$\max(1, T)$$

This prevents:

- negative targets
- unbounded difficulty collapse
- arithmetic overflow

Target always remains within a finite integer range.

5.2 Integer-Based Implementation

Although ASERT is exponential conceptually, implementation uses:

- fixed-point integer arithmetic
- bit-shift approximations

- compact (nBits) conversion
No floating-point operations are used.

Reasons:

1. eliminate cross-node rounding discrepancies
2. ensure platform independence
3. guarantee deterministic consensus

5.3 Small Time-Error Regime

For small Δt , the exponential behaves approximately linearly:

$$2^{\frac{\Delta t}{\tau}} \approx 1 + \frac{\Delta t}{\tau} \ln 2$$

This implies difficulty does not oscillate violently for small timing noise and converges smoothly toward the target interval. Under extreme errors, PowLimit bounds still apply.

6. Anchor Determinism and Time Constraints

6.1 Median Time Past (MTP)

Block time must satisfy the Median Time Past constraint:

$$t_h > \text{MTP}(h - 1)$$

MTP limits abnormally low timestamps and reduces time-warp attack surface.

6.2 Limiting the Impact of Timestamp Manipulation

Even if a single block uses an abnormally high timestamp:

- subsequent blocks are constrained by MTP
- ASERT depends on accumulated timing error
- Target is bounded by PowLimit

Therefore a single manipulated timestamp cannot collapse difficulty instantly; ASERT reacts gradually.

6.3 Consensus Stability Conclusion

Even after adopting ASERT:

1. Target remains in a finite integer range
2. anchor is fixed
3. half-life is constant
4. chain selection remains cumulative-work based

Difficulty computation changes, but the security model (max accumulated work) does not.

7. Consensus Structure of Block Resource Limits

BTCBT enforces explicit resource ceilings to control validation cost and maintain network stability:

1. maximum serialized size (bytes)
2. maximum block weight
3. maximum SigOpsCost

These are consensus constants/parameters. Unlike policy rules, violating them makes a block **invalid**.

7.1 Maximum Serialized Size

For non-consensus-dependent paths (network receive / DoS defense), the maximum serialized size is:

$$\text{MAX_BLOCK_SERIALIZED_SIZE} = 32,000,000 \text{ bytes}$$

This is a compile-time constant. Blocks exceeding this may be rejected before full consensus validation, preventing excessive memory usage.

7.2 Weight Definition

Post-SegWit, block weight is:

$$\text{Weight} = 4 \cdot \text{BaseSize} + \text{WitnessSize}$$

This models witness propagation cost and bounds network resource usage quantitatively.

8. Pre-Fork / Post-Fork Weight Limits

8.1 Pre-Fork Segment

Before the fork, Bitcoin Core's original weight limit applies:

$$\text{MAX_BLOCK_WEIGHT} = 4,000,000$$

This remains unchanged.

8.2 Post-Fork Serialized-Size Cap

After the fork, the serialized-size cap is enforced at consensus level:

$$\text{SerializedSize}(B) \leq 32,000,000$$

Exceeding it makes the block invalid.

8.3 Post-Fork Weight Upper Bound Relation

Post-fork weight must satisfy:

$$\text{Weight}(B) \leq \text{MAX_WEIGHT}_{\text{consensus}}$$

Conceptually:

$$\text{MAX_WEIGHT}_{\text{consensus}} \leq 4 \cdot 32,000,000$$

(This expresses the theoretical ceiling; blocks cannot exceed consensus maximum.)

8.4 Adaptive Block Weight Function

During block-template creation post-fork, a dynamic function may limit the template weight:

$$\text{WeightLimit}_{\text{template}} = f(\text{recent history})$$

But:

$$\text{WeightLimit}_{\text{template}} \leq \text{MAX_WEIGHT}_{\text{consensus}}$$

Adaptive adjustment mitigates overload but does not change the consensus cap.

9. SigOps Limit Model

9.1 SigOpsCost Definition

Each transaction's signature-operation cost is computed as SigOpsCost, quantifying script-validation CPU load.

9.2 Consensus Upper Bound

Consensus constant:

$$\text{MAX_BLOCK_SIGOPS_COST} = 80,000$$

This bound remains post-fork and limits validation CPU usage.

9.3 Handling Violations

A block is invalid if any holds:

1. serialized size cap exceeded
2. weight cap exceeded
3. SigOpsCost cap exceeded

Consensus-violating blocks are not accepted, preventing resource-based DoS vectors.

10. Block Subsidy Function Definition

10.1 Base Subsidy Structure

BTCBT uses a height-based exponentially decaying subsidy. At height h :

$$\text{Subsidy}(h) = R_0 \cdot 2^{-\lfloor \frac{h}{210,000} \rfloor}$$

Where:

- R_0 : initial block subsidy (BTCBT)
- 210,000: halving interval (blocks)
- $\lfloor \cdot \rfloor$: integer floor

Subsidy depends only on height, not time. Reasons:

1. deterministic issuance
2. separation of difficulty adjustment and monetary policy
3. timestamp distortion does not affect supply

10.2 Halving Index

Halving index:

$$k = \left\lfloor \frac{h}{210,000} \right\rfloor$$

Subsidy per halving epoch:

$$\text{Subsidy}_k = R_0 \cdot 2^{-k}$$

Exactly halves each epoch; no unbounded issuance path exists.

10.3 Coinbase Upper-Bound Condition

Total coinbase outputs in block h must satisfy:

$$\sum \text{CoinbaseOutputs} \leq \text{Subsidy}(h) + \text{Fees}(h)$$

Where $\text{Fees}(h)$ is the sum of transaction fees in the block. Violations invalidate the block, mathematically constraining issuance.

Page 11 — 11. Proof of Total Supply Cap

11.1 Cumulative Issuance

Total issuance is an infinite series:

$$\text{Supply}_\infty = \sum_{k=0}^{\infty} (210,000 \cdot R_0 \cdot 2^{-k})$$

Which reduces to a geometric sum:

$$\text{Supply}_\infty = 210,000 \cdot R_0 \cdot \sum_{k=0}^{\infty} 2^{-k} = 210,000 \cdot R_0 \cdot 2$$

Thus:

$$\text{Supply}_\infty = 420,000 \cdot R_0$$

This is the idealized mathematical upper bound.

11.2 Upper-Bound Property

Subsidy satisfies:

$$\lim_{h \rightarrow \infty} \text{Subsidy}(h) = 0$$

Therefore cumulative supply approaches a finite bound.

11.3 Integer Arithmetic Guarantee

Subsidy is computed in integer minimal units (satoshi-like). No floating-point:

- no rounding drift
 - no cumulative issuance error
- Ensures deterministic monetary policy at consensus level.

12. Post-Fork Monetary Policy Invariance

12.1 Subsidy Function Unchanged

The fork changes:

- difficulty adjustment model (ASERT)
 - block resource-limit structure
- But **does not** change the subsidy function. Height-based subsidy definition remains identical.

12.2 No Inflation Path

Additional issuance would require:

1. subsidy-function modification
 2. consensus parameter changes
 3. a majority-agreed hard fork
- No runtime issuance adjustment path exists.

12.3 Separation of Issuance Rate and Total Supply

Difficulty adjustment changes block production speed, but total supply is determined by:

$$\text{Supply}(H) = \sum_{h=0}^H \text{Subsidy}(h)$$

Time affects issuance rate (per unit time), not the total supply cap.

13. Security Model

13.1 PoW-Based Consensus

Bitcoin is SHA-256d PoW. Define per-block work and accumulated work:

$$\text{ChainWork} = \sum \text{Work}(b)$$

Chain selection:

$$\text{BestChain} = \arg \max \text{ChainWork}$$

Lower target implies higher work; the chain with more computation wins.

13.2 Security Assumption

Let attacker hash fraction be q , honest network fraction be p with:

$$p = 1 - q$$

Security holds under:

$$q < \frac{1}{2}$$

If the attacker does not control a majority, the honest chain dominates in accumulated work over time. This matches Bitcoin's SHA-256 security model.

14. Reorganization Probability Model

14.1 Basic Exponential Approximation

For confirmation depth z , a simple approximation for attacker reversal probability:

$$P_{\text{reorg}}(z) \approx \left(\frac{q}{p}\right)^z \text{ (for } q < p)$$

As z increases, reversal probability decreases exponentially.

14.2 Poisson-Based Model (Nakamoto-Style)

Attacker block production is modeled as a Poisson process. Let:

$$\lambda = z \cdot \frac{q}{p}$$

Then the probability that the attacker catches up can be written in the Nakamoto form:

$$P = 1 - \sum_{k=0}^z \frac{e^{-\lambda} \lambda^k}{k!} \left(1 - \left(\frac{q}{p}\right)^{z-k}\right)$$

This follows the probability-based chain competition model from Nakamoto (2008).

14.3 Block Interval and Reorg Probability

Target block interval:

$$T = 300 \text{ seconds}$$

Reorg probability depends primarily on hash ratio q/p , not directly on T . Changing T changes attempts per unit time but does not change the depth-based probabilistic structure.

15. Independence of Difficulty Adjustment and Security

15.1 ASERT vs PoW Structure

ASERT changes target computation but does not change:

- SHA-256d hash function
- work definition
- accumulated work calculation
- chain selection rule

Thus the security model's mathematical structure remains unchanged.

15.2 Timestamp Constraints

Block time follows MTP:

$$t_h > \text{MTP}(h - 1)$$

ASERT consumes timestamp inputs subject to MTP. Timestamp distortion can have gradual effects but cannot instantly collapse difficulty.

15.3 Long-Run Security Structure

BTCBT preserves:

1. max accumulated-work principle
2. height-based consensus transition
3. integer-based difficulty computation
4. exponentially decaying subsidy

16. Fork Activation Boundary

16.1 Height-Based Transition Rule

Fork boundary is defined by a fixed constant:

$$H_f = 903,844$$

Applied rule:

$$\text{pre-fork if } h \leq H_f, \text{post-fork if } h \geq H_f + 1$$

This is defined by `btcbt_fork_block_height`; no runtime change path exists.

16.2 State Definition

Chain state at height h is State_h . Pre-fork maintains:

- UTXO set continuity
- genesis continuity
- SHA-256d definition continuity

BTCBT does not reset state; it transitions continuously.

16.3 Reorg Independence

Even under reorg, applicable rules are determined by height:

- independent of hash distribution
- not vote-based
- height-only

This guarantees mathematical determinism of the activation boundary.

17. Node Validation Determinism

17.1 Block Validation Function

Block validation is a deterministic function:

$$\text{Validate}(B) \in \{\text{TRUE}, \text{FALSE}\}$$

Checks:

1. PoW condition
2. serialized size cap
3. weight cap
4. SigOpsCost cap
5. coinbase output cap
6. UTXO spend validity
7. script evaluation TRUE

Given identical block input and consensus constants, all nodes produce identical results.

17.2 Deterministic Work Computation

Per-block work derives from integer target reconstructed from nBits. No floating point, no platform-dependent path → identical ChainWork across nodes.

17.3 Composition of Consensus Functions

Consensus can be expressed as composition of:

- validation function
 - chain-selection function
- Difficulty adjustment, subsidy, and resource limits are computed deterministically within this structure.

18. UTXO State Transition Model

18.1 State Transition Function

Block-apply function:

$$\text{State}_{h+1} = \text{Apply}(\text{State}_h, B_{h+1})$$

Let:

- $\text{Consumed}(B)$: outputs spent by block B
- $\text{Created}(B)$: outputs created by B

Then:

$$\text{UTXO}_{h+1} = (\text{UTXO}_h \setminus \text{Consumed}(B_{h+1})) \cup \text{Created}(B_{h+1})$$

18.2 State Determinism

For identical State_h and identical block B , the resulting State_{h+1} is identical. Computation is integer-based; no precision loss.

18.3 Double-Spend Prevention

If a block attempts:

- duplicate spends
 - spending non-existent outputs
- the block is invalid. This preserves state consistency.

18.4 Consensus Convergence Condition

Consensus converges if:

1. same genesis
2. same consensus constants
3. same input block sequence

Under these, the network converges to a single chain.

19. Network Protocol Model

19.1 P2P Message Framing

BTCT uses the same P2P framing as Bitcoin Core.

Message header fields:

Field	Size
Magic	4 bytes
Command	12 bytes
Length	4 bytes
Checksum	4 bytes

This framing defines block/transaction propagation format and is unchanged.

19.2 Block Relay Conditions

Nodes relay only blocks that are:

- structurally valid
- within serialized-size limit
- pass consensus validation

Invalid blocks are not relayed, preventing network contamination.

19.3 Orphan-Block Approximation

Let propagation delay be d , block interval T . Orphan probability is approximately proportional to:

$$P_{\text{orphan}} \propto \frac{d}{T}$$

With:

$$T = 300 \text{ seconds}$$

In low-delay environments, orphan probability is bounded.

20. Transaction Validity Model

20.1 Transaction Validity Conditions

A transaction tx must satisfy:

1. all inputs exist in UTXO set
 2. no double-spends
 3. script evaluation TRUE
 4. $\text{sum}(\text{inputs}) \geq \text{sum}(\text{outputs})$
- Violation of any makes the transaction invalid.

20.2 Fee Definition

Fees:

$$\text{Fee}(tx) = \sum \text{Inputs}(tx) - \sum \text{Outputs}(tx)$$

Fees are not new issuance; they are distinct from subsidy.

20.3 Coinbase Maturity

Coinbase outputs must satisfy the maturity rule:

Coinbase spendable only after M blocks

Pre-maturity spending is forbidden; this stabilizes issuance under reorg.

21. Consensus Parameter Fixity

21.1 Consensus Constants

The following do not change at runtime:

- Fork Height = 903,844
- Target Block Time = 300 seconds
- ASERT Half-life = 172,800 seconds
- Max Serialized Size = 32,000,000 bytes
- Max SigOps Cost = 80,000
- Halving Interval = 210,000 blocks
- Initial Reward = 50 BTCBT

These are defined as consensus parameters or compile-time constants.

21.2 Conditions for Change

Changing any consensus constant requires:

1. source-code modification
 2. consensus-based hard fork
 3. majority network upgrade
- No runtime adjustment path exists.

22. Consensus Convergence Structure

Consensus can be expressed as:

- validation function + chain-selection function

Difficulty adjustment, subsidy, and resource limits are computed deterministically inside this structure.

22.1 Invariance Conditions for Convergence

If the following hold, the network converges to a single chain:

1. same genesis
 2. same consensus constants
 3. same validation algorithm
- Under these, consensus splits do not arise structurally.

23. Consensus Invariance Conclusion

BTCBT preserves:

1. height-based fork activation
2. deterministic validation
3. integer-based monetary model
4. bounded difficulty target
5. cumulative-work chain selection

23.1 Monetary Policy Invariance

Subsidy is height-determined. Difficulty adjustment may affect issuance rate per time, but not the total cap.

23.2 Security Model Preservation

ASERT changes target computation but does not change:

- SHA-256d hash function
 - work definition
 - chain-selection rule
- Therefore the PoW chain competition model remains intact.

23.3 Summary

BTCBT:

- inherits the Bitcoin chain up to height 903,844
- fixes activation by height
- computes difficulty deterministically via integer arithmetic
- preserves exponential subsidy decay
- selects the chain by maximum accumulated work

Its consensus, monetary policy, and security model are defined within a mathematically consistent system.