21 Stacker Draw Mechanics — Technical Description

Version: 1.2
Status: Informational / Transparency Document
Audience: Protocol engineers, auditors, technically sceptical users

---

0. Rationale

This document precisely defines the draw mechanism used by 21 Stacker in a way that removes ambiguity, discretion, and trust assumptions. It is written for readers evaluating the system under an adversarial operator model.

---

1. Scope and Assumptions

This document describes the draw mechanism exactly as implemented. It does not rely on informal explanation or assumed operator honesty.

Assumptions:

- Bitcoin consensus behaves as specified
- Cryptographic hash functions behave as specified
- Readers do not trust the operator by default

---

2. Threat Model

The operator is assumed adversarial after draw closure.

The mechanism is designed such that, once the draw closes, **no party retains any degrees of freedom capable of influencing the outcome**.

---

3. System Overview

Each draw proceeds through four irreversible phases:

1. **Commitment** — the operator commits to fixed data prior to closure
2. **Closure** — ticket sales stop and the eligible ticket set is frozen
3. **Anchoring** — a Bitcoin block is selected under a non-discretionary rule
4. **Reveal** — a deterministic calculation selects a winner

Once anchoring occurs, the outcome is mathematically determined.

---

4. Commitment Phase

Prior to draw closure, the system publishes a **pre-commit hash** derived from fixed draw metadata.

Properties:

- Publicly observable
- Immutable once published
- Any modification invalidates verification

Rationale:
This prevents post-hoc manipulation by binding the operator to a fixed input before external randomness is known.

---

5. Closure and Ticket Freezing

At the declared close time:

- Ticket issuance ceases
- All eligible ticket identifiers are collected
- The list is sorted, frozen, and hashed

Only identifiers present in this frozen list are eligible to win.

Rationale:
This eliminates attacks based on ticket insertion, removal, or reordering after closure.

---

6. Bitcoin Anchor Selection

The randomness source is defined as:

> **The first Bitcoin block mined strictly after the draw close timestamp**

Both the block height and block hash are recorded.

Rationale:

- Future blocks are unpredictable
- Selecting the *first* qualifying block removes discretion
- Any flexibility here would permit cherry-picking

---

7. Seed Construction

7.1 Inputs

Two immutable inputs are used:

- Bitcoin anchor block hash (hex-encoded)
- Pre-commit hash (hex-encoded)

7.2 Binary Normalisation

Each hex string is converted into its raw 32-byte binary representation.
If conversion fails, a defensive hash of the hex string is used to ensure a valid byte sequence.

Rationale:
This removes ambiguity arising from string encoding or textual interpretation.

7.3 Concatenation

The byte sequences are concatenated in a fixed order:

```
anchor_hash_bytes || precommit_hash_bytes
```

Rationale:
Order fixation ensures determinism and prevents multiple valid interpretations of the same inputs.

7.4 Final Hashing

The concatenated bytes are hashed using SHA-256, yielding a 32-byte digest.

Rationale:
This evenly mixes both inputs and ensures a uniformly distributed output independent of input structure.

---

8. Hash-to-Integer Mapping

The final hash digest is interpreted numerically as follows:

- The first 8 bytes are selected
- These bytes are read as a big-endian base-256 integer
- The integer is constructed incrementally while applying modulo

This is mathematically equivalent to converting the bytes into a large integer and then reducing it.

Rationale for Using 8 Bytes

SHA-256 output is already uniformly distributed.
For all realistic ticket counts, the entropy contained in the first 8 bytes (64 bits) vastly exceeds what is required for unbiased selection.

Using additional bytes would not change the outcome distribution, while using fewer bytes would reduce safety margins.

Incremental modulo construction avoids integer overflow while remaining mathematically equivalent to full integer reduction.

---

9. Modulo Reduction

```
winner_index = integer mod ticket_count
```

This produces a zero-based index.

Rationale:
Modulo guarantees a uniform mapping into the ticket range without bias or rounding.

---

10. Winner Resolution

The index is applied to the **frozen ticket list**, not to an assumed numeric interval.

Properties:

- Only real tickets are selectable
- Gaps in numbering are irrelevant
- Mapping is bound to the frozen state

---

11. Determinism and Verification

Once the commitment and anchor exist:

- The outcome is uniquely determined
- No alternative execution paths exist
- Independent recomputation must converge on the same result

Appendix A — Independent Reproduction

To independently reproduce a draw result:

1. Obtain the published pre-commit hash
2. Obtain the Bitcoin anchor block hash
3. Convert both from hex to raw bytes
4. Concatenate bytes: `anchor || precommit`
5. Compute SHA-256 of the concatenation
6. Take the first 8 bytes of the digest
7. Interpret as a big-endian integer
8. Compute modulo by the frozen ticket count
9. Index into the frozen ticket list (0-based)

Any compliant implementation must produce the same winner.

---

12. Non-Goals

This mechanism does not attempt to:

- Optimise for minimal explanation complexity
- Obscure internal behaviour
- Require trust in the operator

Its sole objective is to eliminate outcome discretion.

---

13. Summary

The 21 Stacker draw mechanism combines cryptographic commitment, public Bitcoin randomness, and deterministic computation to remove manipulation vectors.

Any perceived complexity exists to constrain choice, not to conceal behaviour.

End of document