OpenRNG Formal Specification

Technical Specification (Draft)
Lotto-Tech Standard for Verifiable Fairness and Randomness

MonkGames

January 19, 2026

1 Introduction

OpenRNG is a lotto-tech standard for verifiable fairness and randomness in lottery systems. It provides infrastructure for operators to demonstrate provably fair draws to regulators and participants.

This specification defines the technical requirements, protocols, and verification procedures for implementing OpenRNG-compliant lottery systems.

2 Scope

This specification covers:

- Cryptographic commitment schemes for draw inputs
- Verifiable randomness generation and sources
- Post-draw verification procedures
- Auditability requirements

OpenRNG does not specify lottery game mechanics, prize structures, or regulatory frameworks. It provides technical infrastructure that operators integrate into their systems.

3 Mathematical Notation

Let $\mathbb{Z}_q$ denote the integers modulo q, where q is a large prime. Let $H : \{0,1\}^* \to \{0,1\}^\lambda$ denote a cryptographic hash function with output length λ (typically SHA-256, so $\lambda = 256$). Let $\mathcal{M}$ denote the message space (draw inputs), and $\mathcal{C}$ denote the commitment space. Let $\mathcal{R}$ denote the randomness space for commitments, and $\mathcal{S}$ the seed space for randomness extraction. Let $\mathcal{O}$ denote the output space of the lottery. Let $t \geq 2$ denote the minimum number of independent seed sources required for randomness extraction.

A commitment scheme consists of three algorithms:

- $\mathsf{Commit}(m, r) \to c$: Commitment algorithm
- $\mathsf{Open}(c, m, r) \to \{0,1\}$: Opening verification algorithm
- $\mathsf{Verify}(c, m, r) \to \{0,1\}$: Public verification algorithm

4 Definitions

Definition 1 (Commitment). *A cryptographic commitment is a pair* (c, r) *where* $c = \mathsf{Commit}(m, r)$ *binds an operator to a message* $m \in \mathcal{M}$ *using randomness* r *before the message or randomness are revealed. The commitment* c *is published, while* r *remains secret until the reveal phase.*

Definition 2 (Verifiable Randomness). *A verifiable randomness source is a function* $R : \mathcal{S}^t \to \{0,1\}^k$ *that maps* $t \geq 2$ *independent seeds* $s_1, s_2, \ldots, s_t \in \mathcal{S}$ *to* k *bits of randomness, where each* s_i *is derived from publicly observable events from independent systems that cannot be manipulated by the operator.*

Definition 3 (Draw). *A draw* D *is a triple* (I, R, O) *where:*

- $I = \{c_1, \ldots, c_n\}$ *is a set of committed inputs*
- $R = R(s_1, s_2, \ldots, s_t)$ *is verifiable randomness extracted from* $t \geq 2$ *independent seed sources*
- $O = f(I, R)$ *is the output computed by deterministic function* f

Definition 4 (Verification). *Verification is the process of confirming that* $\mathsf{Verify}(c_i, m_i, r_i) = 1$ *for all commitments* $c_i \in I$ *with revealed* (m_i, r_i)*, that* R *was correctly extracted from seed sources* $\{s_1, s_2, \ldots, s_t\}$*, and that* $O = f(\{m_1, \ldots, m_n\}, R)$*.*

5 Guarantees

OpenRNG provides the following guarantees:

5.1 Inspectability

Draw processes and verification steps are publicly documented. All procedures, algorithms, and verification steps are available for examination.

5.2 Verifiability

Post-draw outcomes can be independently verified by anyone using public information and standard verification procedures. Verification requires no trust in the operator.

5.3 Auditability

Full audit trail for regulators and technical auditors. All draw inputs, randomness sources, and outputs are documented and verifiable.

6 Boundaries

OpenRNG does not:

- Operate lotteries
- Bypass regulators
- Remove operator responsibility
- Market to consumers

The standard provides technical infrastructure that operators integrate. Operators retain control of business operations, regulatory relationships, and participant relationships.

7 Cryptographic Primitives

7.1 Commitment Scheme

OpenRNG uses a hash-based commitment scheme defined as follows:

Definition 5 (Commitment Algorithm)**.** *For message $m \in \mathcal{M}$ and randomness $r \in \{0,1\}^\lambda$:*

$$\begin{aligned} \mathsf{Commit}(m, r) &= H(m \| r) && (1) \\ &= H(\mathsf{encode}(m) \| r) && (2) \end{aligned}$$

where $\mathsf{encode} : \mathcal{M} \to \{0,1\}^*$ *is an injective encoding function.*

Definition 6 (Opening Algorithm)**.** *Given commitment c, message m, and randomness r:*

$$\mathsf{Open}(c, m, r) = \begin{cases} 1 & \textit{if } \mathsf{Commit}(m, r) = c \\ 0 & \textit{otherwise} \end{cases} \tag{3}$$

Property 1 (Binding)**.** *For all polynomial-time adversaries $\mathcal{A}$:*

$$\Pr\left[\begin{array}{c} (m, r, m', r') \leftarrow \mathcal{A}(1^\lambda) : \\ m \neq m' \wedge \mathsf{Commit}(m, r) = \mathsf{Commit}(m', r') \end{array} \right] \leq \mathsf{negl}(\lambda) \tag{4}$$

where $\mathsf{negl}(\lambda)$ *is negligible in λ.*

Property 2 (Hiding)**.** *For all $m_0, m_1 \in \mathcal{M}$ and uniform randomness r:*

$$\{c : c \leftarrow \mathsf{Commit}(m_0, r)\} \approx_c \{c : c \leftarrow \mathsf{Commit}(m_1, r)\} \tag{5}$$

where $\approx_c$ denotes computational indistinguishability.

7.2 Randomness Extraction

Verifiable randomness is extracted from multiple independent, publicly observable sources using a deterministic extraction function. This multi-source approach ensures unpredictability and resistance to manipulation by any single party.

Definition 7 (Multi-Source Randomness Extraction)**.** *Given $t \geq 2$ independent seed sources $s_1, s_2, \ldots, s_t$ (e.g., block hashes from different blockchains, external randomness beacons, VDF outputs):*

$$R(s_1, s_2, \ldots, s_t) = H(s_1 \| s_2 \| \ldots \| s_t \| \mathsf{nonce}) \tag{6}$$

where nonce *is an optional operator-provided nonce that is committed before any s_i is known, and H is a cryptographic hash function.*

Algorithm: Randomness Generation

1. **Input:** Independent seed sources $\{s_1, s_2, \ldots, s_t\}$ with $t \geq 2$, optional nonce n (if used, must be committed in Phase 1)
2. **Output:** Random value $R \in \{0,1\}^k$
3. **Procedure:**
   - Concatenate seed sources: $S \leftarrow s_1 \| s_2 \| \ldots \| s_t$
   - If nonce n is used: $R \leftarrow H(S \| n)$

- Else: $R \leftarrow H(S)$

4. **Return:** R

Property 3 (Verifiability). *Given seed sources $\{s_1, s_2, \ldots, s_t\}$ and nonce n (if used), any party can compute $R(s_1, s_2, \ldots, s_t)$ independently. No secret information is required.*

Property 4 (Multi-Source Unpredictability). *An adversary controlling at most $t-1$ of the t independent seed sources cannot predict the output randomness R with non-negligible advantage, assuming at least one source is unpredictable and independent.*

7.3 Draw Protocol

Algorithm: OpenRNG Draw Protocol

Phase 1: Commitment

1. For each input $m_i \in \{m_1, \ldots, m_n\}$:
    - Generate random $r_i \xleftarrow{\$} \{0,1\}^\lambda$
    - Compute $c_i \leftarrow \mathsf{Commit}(m_i, r_i)$
    - Publish commitment c_i

Phase 2: Randomness Generation

1. Observe $t \geq 2$ independent seed sources $s_1, s_2, \ldots, s_t$ at designated times $T_1, T_2, \ldots, T_t$
2. Concatenate seed sources: $S \leftarrow s_1 \| s_2 \| \ldots \| s_t$
3. Compute $R \leftarrow H(S \| n)$ where n is committed nonce (if used)
4. Publish all seed sources $\{s_1, s_2, \ldots, s_t\}$, concatenation S, and computed R

Phase 3: Reveal

1. For each commitment c_i:
    - Publish opening (m_i, r_i)
    - Verify $\mathsf{Open}(c_i, m_i, r_i) = 1$

Phase 4: Computation

1. Compute $O \leftarrow f(\{m_1, \ldots, m_n\}, R)$
2. Publish output O

7.4 Verification Algorithm

Algorithm: Verification Procedure

1. **Input:** Draw data: commitments $\{c_1, \ldots, c_n\}$, openings $\{(m_1, r_1), \ldots, (m_n, r_n)\}$, seed sources $\{s_1, s_2, \ldots, s_t\}$, randomness R, output O, optional nonce n
2. **Output:** Verification result $v \in \{0, 1\}$
3. Initialize $v \leftarrow 1$
4. For each commitment c_i with opening (m_i, r_i):
    - If $\mathsf{Open}(c_i, m_i, r_i) \neq 1$, set $v \leftarrow 0$ and break

5. Verify randomness: Concatenate seed sources $S \leftarrow s_1 \| s_2 \| \ldots \| s_t$, then compute $R' \leftarrow H(S \| n)$ (if nonce used, else $R' \leftarrow H(S)$)
   - If $R' \neq R$, set $v \leftarrow 0$ and break
6. Verify output: Compute $O' \leftarrow f(\{m_1, \ldots, m_n\}, R)$
   - If $O' \neq O$, set $v \leftarrow 0$ and break
7. **Return:** v

Theorem 1 (Verification Correctness)**.** *If all parties follow the protocol honestly, the verification algorithm returns $v = 1$ with probability 1.*

Theorem 2 (Soundness)**.** *If the verification algorithm returns $v = 1$, then the output O was computed correctly from the revealed inputs and verified randomness with overwhelming probability, assuming the cryptographic hash function H is collision-resistant.*

8 Implementation Details

8.1 Hash Function Requirements

The commitment scheme and randomness extraction require a cryptographic hash function satisfying:

- Collision resistance: Finding $x \neq x'$ such that $H(x) = H(x')$ is computationally infeasible
- Preimage resistance: Given y, finding x such that $H(x) = y$ is computationally infeasible
- Second preimage resistance: Given x, finding $x' \neq x$ such that $H(x) = H(x')$ is computationally infeasible

Recommended: SHA-256 ($\lambda = 256$) or SHA-3-256.

8.2 Message Encoding

Messages $m \in \mathcal{M}$ must be encoded deterministically before commitment. Encoding function $\mathsf{encode} : \mathcal{M} \to \{0,1\}^*$ must be:

- Injective: $\mathsf{encode}(m) = \mathsf{encode}(m') \implies m = m'$
- Canonical: Same input always produces same output
- Unambiguous: Output can be decoded back to original message

For structured data (JSON, arrays), use canonical serialization (e.g., RFC 8785 JSON Canonicalization Scheme).

8.3 Randomness Requirements

Commitment randomness r_i must be:

- Cryptographically random: $r_i \xleftarrow{\$} \{0,1\}^\lambda$ (uniformly random)
- Independent: Each r_i generated independently
- Secret: Not revealed until Phase 3 (Reveal)
- Unpredictable: Not derivable from public information

Use cryptographically secure pseudorandom number generator (CSPRNG) for generating r_i.

8.4 Seed Source Requirements

Each seed source s_i must be:

- Publicly observable: Anyone can access s_i at designated time T_i
- Unpredictable: Operator cannot predict s_i at commitment time with non-negligible advantage
- Unbiased: No single party can meaningfully influence distribution of s_i
- Timely: Available before randomness generation but after commitment phase
- Independent: Sources must be independent such that controlling one source does not provide information about others

Minimum Requirements:

- At least $t = 2$ independent seed sources must be used
- Sources must be from different systems or time periods to ensure independence
- At least one source must be from a system where the operator has no influence (e.g., external randomness beacon, different blockchain)

Recommended Seed Sources:

- Block hashes from different blockchains (e.g., Bitcoin, Ethereum) at specified future block heights
- External randomness beacons (e.g., NIST Randomness Beacon, drand)
- VDF (Verifiable Delay Function) outputs from publicly committed challenges
- Timestamps from independent time servers with cryptographic attestation

Security Model: The multi-source approach ensures that even if an adversary controls $t - 1$ sources, they cannot predict the final randomness, provided at least one source remains unpredictable and independent.

8.5 Output Computation Function

The function $f : \mathcal{M}^n \times \{0, 1\}^k \to \mathcal{O}$ must be:

- Deterministic: $f(m, R) = f(m, R)$ always
- Public: Algorithm is publicly specified
- Efficient: Computable in polynomial time
- Unbiased: Output distribution is uniform (or specified distribution) when inputs are uniform

For lottery selection: f typically combines inputs with randomness to select winners using specified selection rules (e.g., weighted random selection, tournament bracket).

8.6 Production Implementation

Drawballz is a production lottery game built on OpenRNG standards. It demonstrates OpenRNG requirements in practice.

The Drawballz implementation:

- Uses SHA-256 for commitments and randomness extraction
- Commits player entries before randomness generation
- Extracts randomness from multiple independent sources (e.g., multiple blockchain block hashes from different chains, external randomness beacons)
- Provides public verification tools for post-draw verification
- Maintains full audit trail of all seed sources, commitments, randomness, and outcomes

9 Regulatory Framework

OpenRNG is designed to complement existing regulatory frameworks. It enables independent verification by regulators and provides additional auditability within established regulatory structures.

Operators implementing OpenRNG must continue to comply with all applicable regulatory requirements in their jurisdiction.

10 Version History

- Draft 1.0 – Initial specification with single-source randomness
- Draft 1.1 – January 19, 2026– Updated to multi-source randomness extraction for enhanced security