Dragon Blast

K.A.R. 111-23-21 · eInstant Cert Pack v1

Issued: 2026-05-15T21:40:57.554085Z
Authority: Multi.Bingo iLottery RGS · Operator: Kobow
Signing key: Ed25519 · ID `d651f1b9f793f40d`

Math validation (bit-exact)

Pool size	1,000,000 shares
Winners	590,958
Losers	409,042
Overall odds	1 in 1.69
Σ multipliers	882,377
RTP @ unit denom	88.2377%
Paytable tiers	8
Top prize (max denom)	$150,000
Bonus engine	firework_expander
Jackpot levels	none

1. Game specification (K.A.R. 111-23-21)

Dragon Blast is an eInstant lottery game governed by Kansas Administrative Regulation K.A.R. 111-23-21. The game is operated as a finite-pool ticket-vending model with 1,000,000 shares per pool. The bit-exact paytable below has been validated against the regulator-published spec, with empirical winner counts, sum-of-multipliers, and overall odds matching the regulation to integer precision.

Denominations (Kansas-strict): $1, $2, $3, $4, $5, $10, $20, $30

Prize formula (K.A.R. j): prize_cents = multiplier × denom_cents × tickets_purchased

Bonus engine: firework_expander (deterministic from pool_seed + position; replay endpoint available)

2. Paytable (bit-exact K.A.R.)

Multiplier	Count in pool	Odds (1 in)
×5000	1	1,000,000.00
×1000	39	25,641.03
×150	575	1,739.13
×25	4,807	208.03
×5	25,224	39.64
×2	86,070	11.62
×1	193,142	5.18
×0.5	281,100	3.56
×0 (non-winning)	*409,042*	—

3. Pool architecture

Each pool consists of exactly **1,000,000 shares** arranged in a deterministic shuffle. Pool generation flow:

1. **RNG seed** obtained from the certified RGS service `multibingo-rng:9443` (RNG_STRICT=true, no local fallback) per K.A.R. RNG requirements.
2. **Server seed** hashed with SHA-256 → `shuffle_seed_hash` (pre-commit, published before any ticket sells).
3. Paytable counts (table in section 2) are written into the position array, then shuffled with **mulberry32 PRNG** seeded from the SHA-256 of the server_seed. The Fisher-Yates shuffle is deterministic — any auditor with the seed can reproduce the same pool order.
4. The pool's **audit_hash** (Ed25519-signed envelope of game_code, variant_id, jurisdiction, pool_size, paytable_hash, deterministic_seed_hash, prev_audit_hash) chains into the audit history.
5. Ticket positions are consumed sequentially as players buy; each ticket's outcome is the multiplier at its position in the shuffled array.

4. Audit chain

Per K.A.R. (h) compliance, every day at end-of-day the system seals a **daily Merkle root** over all tickets sold + all pools opened that day. Leaves are computed as:

```
 ticket_leaf =
sha256(id|pool_id|position|outcome_multiplier|prize_cents|validation_hash)
 pool_leaf = sha256('POOL'|id|game_code|variant_id|jurisdiction|shuffle_seed_hash|det
erministic_seed_hash|audit_hash)
```

The root is signed with the operator's Ed25519 private key (key ID `d651f1b9f793f40d`) and persisted in an immutable PostgreSQL row (WORM-enforced via BEFORE UPDATE/DELETE trigger). The audit blob with all leaves is also written to disk and (in production) replicated to S3 with Object Lock and 3.5-year retention. Each day's row stores `prev_root_sha256` forming a hash chain back to genesis.

5. Verifier surface (public REST, CORS *, no auth)

Endpoint	Purpose
`GET /api/ilottery/verify/:ticket_id`	Replay outcome from pool_seed + position; returns match boolean
`POST /api/ilottery/vector/replay`	Recompute outcome from paytable + seed (no DB lookup; for cert)
`GET /api/ilottery/reveal/:pool_id`	Pool seed reveal (closed pools or sandbox)
`GET /api/ilottery/merkle/days`	List recent daily Merkle anchors
`GET /api/ilottery/merkle/day/:date`	Get specific anchor (root + Ed25519 sig + audit blob path)
`GET /api/ilottery/merkle/proof/:ticket_id`	Merkle inclusion proof (sibling-hash path + verifies bool)
`GET /api/ilottery/audit/public-key`	Ed25519 public key (PEM + JWK) for offline verification
`POST /api/ilottery/audit/verify-signature`	Server-side verify {message, signature_hex}
`GET /api/ilottery/audit/summary`	System-wide stats (games/tickets/jackpot/anchors)

6. Test vectors v1

URL: `https://kobowlotto.com/mathforge/ilottery/gdd/dragon_blast/vectors_v1.json`
Contents: 6 tiers × ~100 vectors each (TOP / MID_HIGH / MID_MID / MID_LOW / X1 / LOSER). Per-vector fields:
`{id, tier, pool_seed_hex, position, expected_multiplier, expected_prize_cents{per denom}}`

Verification protocol: POST /api/ilottery/vector/replay with {game_code, pool_seed_hex, position} returns expected_multiplier. Auditors can independently compute the same outcome by running mulberry32(sha256(seed)) Fisher-Yates over the paytable.

7. Operator Ed25519 public key

Algorithm: Ed25519 · **Key ID:** `d651f1b9f793f40d`

```
-----BEGIN PUBLIC KEY-----
MCowBQYDK2VwAyEAU2k0zXnm09QHDuKLSEQYUAFeACAEOTUceEuiRGLSiXs=
-----END PUBLIC KEY-----
```

8. Per-jurisdiction matrix

Jurisdiction	max_tickets	cells	denoms (¢)	cap ($)	cost_formula
BR-charity	1	10	50, 100, 200, 300, 500, 1000, 2000, 3000	$50,000	tickets * denom
GLI-SANDBOX	1	10	100, 200, 300, 400, 500, 1000, 2000, 3000	$50,000	tickets * denom
US-MN-charity	1	10	25, 50, 100, 200, 500, 1000	$50,000	tickets * denom
US-WV	1	10	100, 200, 300, 400, 500, 1000, 2000, 3000	$50,000	tickets * denom

9. Document attestation

This cert pack reflects the system state at issuance time. Live values (jackpot amounts, ticket counts, Merkle roots) can be queried in real time via the verifier endpoints above. The Ed25519 signature on the document SHA-256 attests document integrity; the operator's public key (section 7) is the same key used to sign all daily Merkle anchors and pool audit hashes.