

VIP Riches

K.A.R. 111-23-11 · eInstant Cert Pack v1

Issued: 2026-05-15T21:40:56.756797Z

Authority: Multi.Bingo iLottery RGS · Operator: Kobow

Signing key: Ed25519 · ID d651f1b9f793f40d

Math validation (bit-exact)

Pool size	1,000,000 shares
Winners	292,447
Losers	707,553
Overall odds	1 in 3.4194
Σ multipliers	860,550
RTP @ unit denom	86.0550%
Paytable tiers	28
Top prize (max denom)	\$50,000
Bonus engine	pick_n_cases
Jackpot levels	none

1. Game specification (K.A.R. 111-23-11)

VIP Riches is an instant lottery game governed by Kansas Administrative Regulation K.A.R. 111-23-11. The game is operated as a finite-pool ticket-vending model with 1,000,000 shares per pool. The bit-exact payable 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, \$5

Prize formula (K.A.R. j): $\text{prize_cents} = \text{multiplier} \times \text{denom_cents} \times \text{tickets_purchased}$

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

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

Multiplier	Count in pool	Odds (1 in)
x10000	1	1,000,000.00
x3500	1	1,000,000.00
x3250	2	500,000.00
x3000	2	500,000.00
x2500	2	500,000.00
x330	15	66,666.67
x300	15	66,666.67
x250	25	40,000.00
x150	100	10,000.00
x125	60	16,666.67
x60	125	8,000.00
x50	99	10,101.01
x37	200	5,000.00
x35	300	3,333.33
x25	2,000	500.00
x16	1,000	1,000.00
x13	250	4,000.00
x12	4,500	222.22
x10	5,500	181.82
x8	11,000	90.91
x7	14,000	71.43
x6	1,000	1,000.00
x5	1,500	666.67
x4	23,000	43.48
x3	750	1,333.33
x2	22,000	45.45
x1.5	80,000	12.50
x1	125,000	8.00
x0 (non-winning)	707,553	—

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|deterministic_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 payable + 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/vip_riches/vectors_v1.json

Contents: 6 tiers x ~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 payable.

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	6	4	25, 50, 100, 200, 500	\$240,000	tickets * cells * denom
GLI-SANDBOX	6	4	100, 200, 500	\$300,000	tickets * denom
US-MN-charity	6	4	10, 25, 50, 100	\$200,000	tickets * cells * denom
US-WV	6	4	100, 200, 500	\$500,000	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.