Provably Fair System for Pack Opening - Compliance Documentation

Overview

The ProvablyFairSignedSystem ensures that pack opening outcomes are fair, transparent, and verifiable. All item selections during pack opening are determined by cryptographic randomness that can be independently verified by players and auditors.

System Components

1. Server Seed

- **Generation**: Cryptographically secure random 64-byte seed (128 hex characters)
- **Storage**: Encrypted using AES-256-GCM before storage in database
- **Hash**: SHA-512 hash is publicly disclosed before use (allows verification after reveal)
- **Rotation**: Server seeds automatically rotate every 60 seconds
- **Source**: Latest blockchain block hash from the WinnablesPacks contract chain

2. Client Seed

- **Source**: Provided by the user or generated and stored for each user
- **Storage**: Stored in database linked to user account
- **Purpose**: Ensures users can influence randomness through their choice

3. Blockchain Block Hash

- **Source**: Latest block hash from the blockchain at time of seed generation
- **Purpose**: Provides additional entropy from external source (blockchain)
- **Verification**: Publicly verifiable on blockchain

4. Nonce

- **Format**: Timestamp-based unique identifier for each pack opening
- **Purpose**: Ensures each opening produces a unique outcome even with same seeds

Pack Opening Process

Step 1: Outcome Generation

When a pack is opened, the system generates a verifiable outcome number using:

```
Combined String = BlockHash + ClientSeed + Nonce + ServerSeed
Hash = SHA-512(Combined String)
Outcome Number = First 8 hex characters of hash (modulo 1,000,000)
```

This produces a random number between 0 and 999,999 that is deterministic and verifiable.

Step 2: Item Selection

The outcome number is used to select an item from the pack based on probability:

1. Pack items are sorted by probability (lowest to highest)
2. Cumulative probability ranges are calculated for each item
3. The item whose cumulative probability range contains the outcome number is selected
4. Items with lower probabilities (rarer items) are checked first

Example:

- Item A: Probability 50,000 (5%)
- Item B: Probability 100,000 (10%)
- Item C: Probability 850,000 (85%)

If outcome = 75,000:

- Cumulative check: 50,000 < 75,000 < 150,000 → **Item B selected**

Step 3: Response to Client

The system returns:

- Selected item details
- Outcome number
- Server seed hash (for verification)
- Nonce
- Timestamp
- Verification data (clientSeed, nonce, serverSeedHash)

Verification Process

Players can verify fairness by:

1. **Before opening**: Request and store the server seed hash
2. **After opening**: Receive the outcome and verification data
3. **After seed rotation**: Server seed is revealed (decrypted)
4. **Verification**: Recalculate the hash using:
   - BlockHash (from blockchain)
   - ClientSeed (provided by player)
   - Nonce (from opening)
   - ServerSeed (now revealed)

If the calculated outcome matches the provided outcome, the result was fair.

Security Features

1. **Server Seed Encryption**: Server seeds are encrypted at rest (AES-256-GCM)
2. **Pre-commitment**: Server seed hash is stored before use, preventing manipulation
3. **Blockchain Entropy**: Uses publicly verifiable blockchain block hash
4. **Automatic Rotation**: Seeds rotate every 60 seconds, limiting exposure window
5. **Deterministic Selection**: Outcome is mathematically derived, not chosen

Database Storage

- **ProvablyFairSeed**: Stores server seed hash, encrypted seed, block hash, block number
- **UserSeed**: Stores client seed for each user
- **Active Flag**: Only one seed is active at a time

Compliance Notes

- All outcomes are deterministic and mathematically verifiable
- Server cannot manipulate results after server seed hash is disclosed
- Probability distributions are enforced: total must equal 1,000,000

- Expected payout is validated against pack price (prevents negative house edge)
- Each opening event produces a unique, traceable outcome

Technical Implementation

Files:

- `src/services/ProvablyFairSignedSystem.ts` - Core provably fair logic
- `src/services/PackOpeningService.ts` - Pack opening with provably fair integration
- `src/services/PackService.ts` - Item selection based on outcome
- `src/controllers/pack.ts` - API endpoints for pack opening

Key Methods:

- `generateGameOutcome()` - Generates verifiable outcome number
- `selectItemFromPack()` - Selects item based on outcome and probabilities
- `rotateSeeds()` - Automatically rotates server seeds