RION Oracle Integration

Overview

This document describes the integration of RION Oracle as a backup oracle service for the Smart Money Tracker platform.

What is RION Oracle?

RION Oracle is a decentralized oracle network built specifically for BNB Chain that provides secure, real-time data feeds for smart contracts. It supports:

- **Price Feeds**: Real-time cryptocurrency price data
- **Prediction Markets**: Outcome verification
- **Gaming Applications**: On-chain game data
- **Custom Data Requests**: Flexible data sourcing

Key Features

- 🔐 **BLS Signature Aggregation**: Efficient data verification
- ⚖️ **Dispute Resolution**: Comprehensive system for data integrity
- ⚡ **Fast Setup**: Deploy in under 30 minutes
- 🛡️ **Secure**: Built with security and reliability at its core
- 🧪 **Testnet Ready**: Full BNB testnet support

Current Implementation Status

⚠️ Placeholder Implementation

The current integration is a **placeholder** implementation because:

1. RION Oracle's official npm/SDK package is not yet publicly available
2. Their SDK documentation is still under development
3. The API endpoints are not yet publicly accessible

When SDK is Available

Once RION releases their official SDK, update the implementation in `/lib/rion-oracle-client.ts`:

```
// Future implementation example:
import { RionClient } from '@rion/sdk'; // When available

const rion = new RionClient({
  apiKey: process.env.RION_API_KEY,
  network: 'bnb-mainnet', // or 'bnb-testnet'
});

const prices = await rion.getPrices(['BTC', 'ETH', 'BNB']);
```

Configuration

Environment Variables

Add to `/app/.env`:

```
# RION Oracle Configuration
RION_API_KEY=your_api_key_here
RION_API_ENDPOINT=https://api.rion-oracle.com  # Optional, will use default
```

Getting RION API Key

1. Visit RION Oracle (https://www.rion-oracle.com)
2. Sign up for an account
3. Navigate to API Keys section
4. Generate a new API key
5. Add to your `.env` file

Usage

Check if RION is Configured

```
import rionOracle from '@/lib/rion-oracle-client';

if (rionOracle.isConfigured()) {
  console.log('RION Oracle is configured');
}
```

Fetch Prices

```
import { fetchRionPrices } from '@/lib/rion-oracle-client';

const result = await fetchRionPrices(['BTC', 'ETH', 'BNB']);

if (result.success) {
  console.log('Prices:', result.data);
  console.log('Provider:', result.metadata.provider);
  console.log('Network:', result.metadata.network);
}
```

Get Single Price

```
import { getRionPrice } from '@/lib/rion-oracle-client';

const btcPrice = await getRionPrice('BTC');

if (btcPrice) {
  console.log(`BTC Price: $${btcPrice.price}`);
  console.log(`Confidence: ${btcPrice.confidence}%`);
}
```

Check Status

```
import { getRionStatus } from '@/lib/rion-oracle-client';

const status = await getRionStatus();

console.log('Configured:', status.configured);
console.log('Available:', status.available);
console.log('Message:', status.message);
console.log('Network:', status.network);
```

Integration with Price Service

RION Oracle is integrated into the price service architecture as a backup source:

Priority Order

1. **CoinGecko** (Primary)
2. **CoinCap** (Fallback 1)
3. **Binance** (Fallback 2)
4. **RION Oracle** (Backup) ← NEW

The price service will automatically try RION Oracle if other sources fail.

Architecture

```
┌─────────────────────┐
│  Price Service      │
│  (lib/price-service)│
└──────────┬──────────┘
           │
           ├─► CoinGecko (Primary)
           ├─► CoinCap (Fallback 1)
           ├─► Binance (Fallback 2)
           └─► RION Oracle (Backup) ← NEW
```

API Endpoints

Fetch Prices

Endpoint: `POST /v1/prices`

Headers:

```
{
  "Content-Type": "application/json",
  "Authorization": "Bearer YOUR_API_KEY"
}
```

Body:

```
{
  "symbols": ["BTC", "ETH", "BNB"]
}
```

Response:

```
{
  "prices": [
    {
      "symbol": "BTC",
      "price": 43250.00,
      "timestamp": 1700000000000,
      "confidence": 99.5,
      "source": "aggregated"
    }
  ],
  "network": "BNB Chain"
}
```

Benefits of RION Oracle

Why Add RION as Backup?

1. **BNB Chain Native**: Specifically built for BNB Chain
2. **High Reliability**: Dispute resolution ensures data integrity
3. **Additional Data Source**: Increases redundancy
4. **Future-Ready**: Will support more data types (predictions, gaming)

Comparison

Feature	CoinGecko	RION Oracle
Rate Limits	30 calls/min (free)	TBD
Chains	Multi-chain	BNB Chain focus
Data Types	Prices only	Prices + more
Verification	Trust-based	Cryptographic proofs
Disputes	No	Yes

Resources

- **Website**: https://www.rion-oracle.com
- **SDK Docs**: https://www.rion-oracle.com/sdk
- **API Docs**: https://www.rion-oracle.com/api-docs
- **GitHub**: https://github.com/rionoracle
- **Twitter**: https://x.com/rionoracle
- **Contact**: hello@rion-oracle.com

Troubleshooting

API Key Not Working

1. Verify the key in `.env` file
2. Check if the key has been activated
3. Ensure you're using the correct environment (testnet vs mainnet)

Connection Errors

1. Check internet connectivity
2. Verify `RION_API_ENDPOINT` is correct
3. Check if RION services are operational

No Data Returned

1. RION may not support the requested symbols yet
2. Network might not be available
3. Check the console logs for detailed error messages

Next Steps

1. ✅ Placeholder client created
2. ⏳ Wait for RION SDK release
3. ⏳ Update client with official SDK
4. ⏳ Obtain RION API key
5. ⏳ Test integration thoroughly
6. ⏳ Monitor performance and reliability

Update Instructions

When RION releases their SDK:

1. Install the SDK:
   ```bash
   cd /home/ubuntu/smart_money_tracker/app
   yarn add @rion/sdk  # Or whatever the package name is
   ```
2. Update `/lib/rion-oracle-client.ts` with official SDK implementation
3. Test the integration:
   ```bash
   yarn test rion-oracle
   ```
4. Update this documentation with actual API details

License

This integration follows the same license as the Smart Money Tracker project.

Last Updated: November 19, 2025
Status: Placeholder Implementation
Next Review: When RION SDK is released