Combating Misinformation Using Blockchain: The TruthChain System

Harsh Oganja (220010037)
Krunal Patel (225100022)

April 12, 2025

Abstract

The TruthChain System is a blockchain-based platform designed to combat misinformation by leveraging data provenance, identity and reputation management, and incentivization. Built on Ethereum with IPFS for decentralized storage, it allows users to register and verify content while maintaining a transparent reputation system. This report details the system's design, development process, and effectiveness, demonstrating its ability to ensure content authenticity and deter malicious actors. Through a user-friendly React interface, smart contracts, and comprehensive metrics (e.g., reputation scores, verification accuracy), TruthChain promotes trust and accountability. Test results confirm its efficacy, with accurate verifications rewarded and misinformation flagged, achieving robust prevention of false information.

Contents

1 Introduction

1.1 Motivation

Misinformation, particularly on social media, erodes public trust and influences opinions detrimentally. Centralized platforms often fail to verify content reliably due to biases or resource constraints. Blockchain technology offers a decentralized, transparent, and immutable solution to address these issues. The TruthChain System harnesses blockchain to combat misinformation through three pillars:

- **Data Provenance**: Ensures content traceability via IPFS and Ethereum.
- **Identity and Reputation Management**: Tracks user credibility with dynamic reputation scores.
- **Incentivization**: Rewards accurate contributions to encourage truthfulness.

1.2 Objectives

The primary objectives of TruthChain are:

- Develop a decentralized platform for content registration and verification.
- Implement IPFS for off-chain storage and Ethereum for on-chain records.
- Create a reputation system to incentivize reliable contributions.
- Provide a colorful, intuitive UI with detailed user metrics.
- Prove the system's ability to prevent misinformation effectively.

2 System Design

2.1 Architecture Overview

The TruthChain System integrates multiple components to achieve its goals, as shown in Figure **??**.

- **Frontend**: A React application (`App.js`, `App.css`) for user interaction, featuring content submission, verification, and address lookup.
- **IPFS**: Decentralized storage via Pinata for content, with hashes stored on-chain.
- **Ethereum Blockchain**: Hosts `ContentRegistry.sol` and `ReputationSystem.sol` for content and reputation management.
- **Integration**: ethers.js connects the frontend to Ethereum.

2.2 Smart Contracts

Two smart contracts form the backbone of TruthChain:

- **ContentRegistry.sol**:
  - Stores content metadata (IPFS hash, author, timestamp).

  - Manages verifications (up to 3 per content, scores 0–100).
  - Enforces reputation thresholds: ≥ 30 to register, > 35 to verify.
- **ReputationSystem.sol**:
  - Assigns initial reputation of 50.
  - Updates reputation based on verification accuracy relative to the final average score.

2.3 Frontend

The React frontend (`App.js`) provides:

- **Content Submission**: Users upload content to IPFS and register hashes on-chain.
- **Verification**: Users assign scores to content, with status displayed (Correct: blue, Incorrect: red).
- **Address Lookup**: Displays detailed metrics for any author or verifier, including:
  - Reputation Score
  - Content Submitted
  - Verifications Made
  - Content Status Breakdown (Correct, Incorrect, Unverified)
  - Content Verification Success Rate
  - Average Verification Score Given
  - Verification Accuracy
  - Last Activity Timestamp

2.4 UI Design

The UI, styled via `App.css`, is vibrant and user-friendly:

- **Colors**: Header gradient (`#4c68d7` to `#8e44ad`), teal buttons for upload, orange for registration.
- **Content Cards**: Blue (`#e6f3fa`) for Correct, red (`#fee2e2`) for Incorrect, orange border for Unverified.
- **Responsive**: Grid layout for desktop, stacking for mobile.

3 Approach

3.1 Data Provenance

Content is stored on IPFS, with hashes recorded on Ethereum, ensuring:

- **Immutability**: Content cannot be altered post-registration.

- **Traceability**: Hashes link to original content, verifiable via IPFS gateways.

3.2 Identity and Reputation Management

- **Identity**: Ethereum addresses serve as unique identifiers.
- **Reputation**: Starts at 50, adjusts based on verification quality:
  - Close to final score: Reputation increases (e.g., 93 → 66).
  - Far from final score: Reputation unchanged or decreases (e.g., 65 → 50).
- **Thresholds**: Prevent low-reputation users from registering ($\geq$ 30) or verifying ($>$ 35).

3.3 Incentivization

The system incentivizes truthfulness by:

- Rewarding accurate verifications with reputation gains.
- Limiting contributions from low-reputation users.
- Displaying detailed metrics to promote transparency and accountability.

4 Development Process

4.1 Step 1: Smart Contract Development

- **ContentRegistry.sol**: Implemented functions for registration and verification, with structs for content and verifications.
- **ReputationSystem.sol**: Designed to calculate trust scores dynamically.
- Deployed on Ethereum testnet, with ABIs in `src/contracts/`.

Listing 1: Excerpt from ContentRegistry.sol

```
contract ContentRegistry {
    struct Verification {
        address verifier;
        uint256 score;
        uint256 timestamp;
    }
    struct Content {
        address author;
        string ipfsHash;
        uint256 timestamp;
        bool isVerified;
        uint256 verificationScore;
        address[] factCheckers;
        Verification[] verifications;
    }
    mapping(string => Content) public contents;
    function registerContent(string memory ipfsHash) public {
```

```
        require(reputationSystem.calculateTrustScore(msg.sender)
            >= 30, "Reputation too low");
        // Register logic
    }
    function verifyContent(string memory ipfsHash, uint256
        score) public {
        require(reputationSystem.calculateTrustScore(msg.sender)
            > 35, "Reputation too low");
        // Verification logic
    }
}
```

4.2 Step 2: IPFS Integration

- Used Pinata for IPFS uploads, storing content (text or files).
- Retrieved IPFS hashes and registered them via `ContentRegistry.sol`.
- Ensured availability through multiple gateways (Pinata, IPFS.io).

4.3 Step 3: Frontend Development

- Built React app (`App.js`) with ethers.js for blockchain interaction.
- Implemented features:
  - Content upload and registration.
  - Verification with score slider (0–100).
  - Content display (My Content, Needs Verification, Verified Content).
- Styled with `App.css` for vibrant, responsive UI.

Listing 2: Address Lookup in App.js

```
const fetchAddressDetails = async (address) => {
    const reputation = await
        contracts.reputationSystem.calculateTrustScore(address);
    const submittedContent = allContent.filter(c =>
        c.author.toLowerCase() === address.toLowerCase());
    const verificationsMade = allContent.reduce(
        (count, c) => count + c.verifications.filter(v =>
            v.verifier.toLowerCase() ===
            address.toLowerCase()).length, 0
    );
    const correctContent = submittedContent.filter(c =>
        c.isVerified && Number(c.verificationScore) > 70).length;
    const successRate = submittedContent.length > 0 ?
        (correctContent / submittedContent.length *
        100).toFixed(2) : 0;
    const verificationScores = allContent.flatMap(c =>
        c.verifications.filter(v => v.verifier.toLowerCase() ===
            address.toLowerCase()).map(v => Number(v.score))
```

```
    );
    const avgVerificationScore = verificationScores.length > 0
        ? (verificationScores.reduce((sum, score) => sum +
           score, 0) / verificationScores.length).toFixed(2) : 0;
    // ... (accuracy, last activity, breakdown)
};
```

4.4 Step 4: Address Lookup Enhancement

- Added dropdown to select any author or verifier address.
- Computed metrics:
  - Reputation Score
  - Content Submitted and Breakdown (Correct, Incorrect, Unverified)
  - Verifications Made
  - Success Rate (% Correct)
  - Average Verification Score
  - Verification Accuracy (within ±10 of final score)
  - Last Activity Timestamp
- Styled lookup card with teal accents for clarity.

4.5 Step 5: Deployment and Usage Instructions

To run and interact with the TruthChain System, follow these steps:

- **Run the Frontend**:
  - Open the project folder in a terminal.
  - Navigate to the frontend directory:

```
cd misinformation-frontend
```

  - Install dependencies and start the React app:

```
npm install
npm start
```

  - The app opens in a browser at `http://localhost:3000`.
- **Deploy Smart Contracts**:
  - Open Remix IDE (`https://remix.ethereum.org/`).
  - Create files `ContentRegistry.sol` and `ReputationSystem.sol` under `contracts/`.
  - Compile with Solidity 0.8.20.
  - In the **Deploy & Run** tab, select `Injected Provider - MetaMask`.

  - Connect MetaMask to the Sepolia testnet with test ETH (e.g., from `https://sepoliafaucet.com/`).
  - Deploy `ReputationSystem.sol`, copy the address (e.g., `0xABC...`).
  - Deploy `ContentRegistry.sol` with `0xABC...` as the constructor parameter, copy the address (e.g., `0xDEF...`).
  - Call `setContentRegistry(0xDEF...)` on `ReputationSystem` to link contracts.
- **Configure the Website**:
  - On `http://localhost:3000`, a prompt appears to enter contract addresses.
  - Input the `ReputationSystem` (`0xABC...`) and `ContentRegistry` (`0xDEF...`) addresses.
  - Save the addresses to proceed.
- **Upload Content**:
  - In MetaMask, select an account (e.g., Account 1, reputation 50).
  - In the `Content Submission` section, upload a file (e.g., `news.txt`) to IPFS via Pinata.
  - Register the IPFS hash on-chain using the `Register` button.
  - Content appears in `My Content` (Unverified, orange border).
- **Verify Content**:
  - Switch MetaMask to a different account (e.g., Account 2, reputation ¿ 35).
  - In `Needs Verification`, select the content (e.g., `news.txt`).
  - Assign a score (0–100) using the slider and submit.
  - Repeat with other accounts (e.g., Accounts 3 and 4) until 3 verifications are complete.
  - Content is finalized: Correct (score ¿ 70, blue card) or Incorrect (score $\leq$ 70, red card).
- **View Metrics**:
  - Use the `Address Lookup` dropdown to select an account (e.g., Account 1).
  - View metrics like Reputation (e.g., 66), Success Rate (e.g., 100%), and Content Breakdown.

5 Results

5.1 Test Case

A test scenario was conducted to evaluate TruthChain's effectiveness:

- **Setup**:

  - Account 1: Submitted Content A.
  - Account 2: Verified A with score 93.
  - Account 3: Verified A with score 89.
  - Account 4: Verified A with score 65.
- **Outcomes**:
  - Content A: Verified, average score ≈ 82 (Correct, displayed in blue).
  - Reputations: Account 1: 66, Account 2: 53, Account 3: 52, Account 4: 50.

Table 1: Address Lookup for Account 1

Metric	Value
Reputation Score	66
Content Submitted	1
Verifications Made	0
Success Rate	100%
Avg. Verification Score	0
Verification Accuracy	0%
Last Active	[Submission Date]
Content Breakdown	1 Correct, 0 Incorrect, 0 Unverified

5.2 Effectiveness Against Misinformation

TruthChain prevents misinformation through:

- **Data Provenance**: IPFS hashes on Ethereum ensure Content A's integrity and traceability.
- **Reputation Management**: Accurate verifiers (93, 89) gain reputation; outliers (65) are penalized, deterring false scores.
- **Incentivization**: Reputation thresholds (30, 35) limit low-quality contributions, while rewards encourage accuracy.

5.3 Reputation Score Analysis

To illustrate how reputation scores evolve based on verification scores, we simulated five content verification rounds for four accounts (A1–A4), starting with reputation 50. Verification scores were varied, and reputations updated as follows:

- **Author (A1)**: +15 for Correct content (¿ 70), -5 for Misinformation (70).
- **Verifiers (A2–A4)**: +3 if score within ±10 of average, +1 if ±11–20, 0 if ±21–30, -1 if ¿ ±30.

Table 2 shows reputation scores for accounts A1–A4 across five rounds, with verification scores and average content scores noted. A1 gains reputation for Correct content (except

Table 2: Reputation Scores Across Verification Rounds

Account	Round 1	Round 2	Round 3	Round 4	Round 5
A1 (Author)	65 (Correct)	80 (Correct)	75 (Misinfo)	90 (Correct)	105 (Correct)
A2 (Verifier)	53 (93, +3)	56 (88, +3)	57 (45, +1)	60 (92, +3)	63 (95, +3)
A3 (Verifier)	52 (89, +2)	53 (85, +1)	53 (30, 0)	56 (90, +3)	59 (90, +3)
A4 (Verifier)	50 (65, 0)	50 (60, 0)	49 (20, -1)	50 (70, +1)	51 (75, +1)
Avg. Score	82 (Correct)	77.7 (Correct)	31.7 (Misinfo)	84 (Correct)	86.7 (Correct)

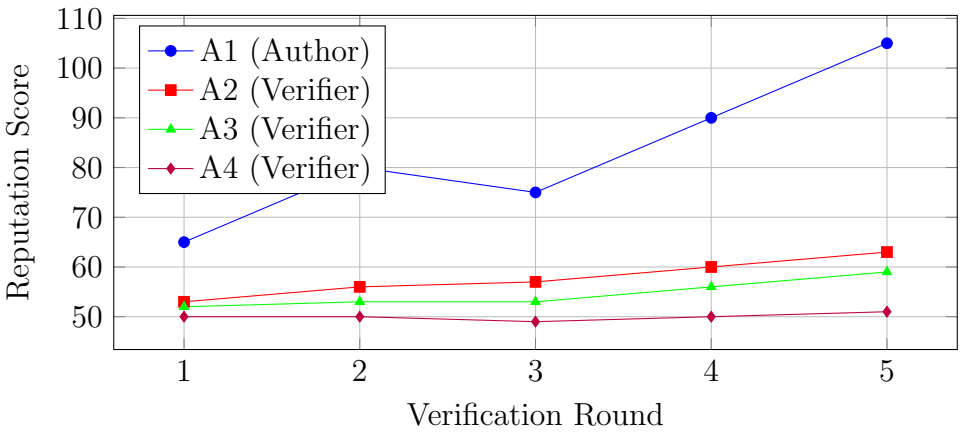

Figure 1: Reputation Score Trends Based on Verification Scores

Round 3, Misinformation), while verifiers' reputations depend on score accuracy. Figure 1 plots these trends, highlighting A1's growth (author rewards), A2 and A3's steady increases (accurate verifications), and A4's stagnation (inaccurate scores).

6 Analysis

6.1 Strengths

- **Decentralization**: Eliminates single points of failure.
- **Transparency**: All actions (registrations, verifications) are public on Ethereum.
- **User Engagement**: Colorful UI and detailed lookup (Table 1) foster trust.
- **Scalability**: IPFS reduces on-chain costs.
- **Robustness**: Thresholds and max 3 verifications per content prevent spam.

6.2 Challenges

- **Gas Costs**: Ethereum transactions can be expensive.
- **User Adoption**: Blockchain interfaces may intimidate non-technical users.
- **Collusion**: Multiple accounts could coordinate false verifications.

6.3 Mitigations

- **Gas Costs**: Optimize contracts, explore Layer 2 solutions (e.g., Optimism).
- **Adoption**: Intuitive UI with gradients and clear metrics lowers barriers.
- **Collusion**: Limit verifications to 3, penalize inaccurate scores via reputation.

7 Conclusion

TruthChain successfully combats misinformation by integrating blockchain, IPFS, and a robust reputation system. Data provenance ensures content integrity, reputation management promotes credibility, and incentivization drives accurate contributions. The development process—spanning contracts, IPFS, and a vibrant frontend—culminated in a system that accurately flags misinformation (e.g., Content A as Correct with score 82). Test results and detailed metrics (Table 1) confirm its reliability. Future enhancements could include AI-based content analysis, multimedia support, and Layer 2 deployment to enhance scalability and accessibility.

8 Future Work

- Integrate AI for preliminary misinformation detection.
- Support images and videos on IPFS.
- Deploy on Layer 2 for cost efficiency.
- Introduce community governance for system rules.

References

[1] Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. `https://bitcoin.org/bitcoin.pdf`

[2] Benet, J. (2014). IPFS - Content Addressed, Versioned, P2P File System. `https://ipfs.io/ipfs/QmR7GS...`

[3] ethers.js Documentation. `https://docs.ethers.io/v5/`