Fake News Detection DApp

CS 765 - Introduction of Blockchains, Cryptocurrencies, and Smart Contracts

Spring 2024

Indian Institute of Technology Bombay

Authors:

Debdoot (23M0765)

Nilava Sarkar (22M0753)

Niraj Jaiswal (22M0779)

Lecturer:

Prof. Vinay Joseph Ribeiro

Dept. of CSE, IIT Bombay

April 18, 2024

1. Introduction

A Decentralized App (DApp) is an application running on a permissionless blockchain. A DApp is usually implemented as a smart contract. A smart contract is essentially a program whose code is on the blockchain. The code is initially put on the blockchain in a transaction. The smart contract can have many functions. Different functions can be invoked by other transactions later on, provided the person invoking the function(s) has the permissions to do so as specified by the smart contract. When different functions are executed, the state of the smart contract are modified.

2. Fake-News Validation

In today's digital landscape, combating misinformation and fake news is paramount. Traditional fact-checking methods often fall short in the face of decentralized dissemination. To address this, we propose a blockchain-based decentralized application (DApp) to combat fake news. By leveraging blockchain's transparency and immutability, users can collectively verify news items, fostering trust in the digital ecosystem. Our DApp empowers users to submit and flag news items, ensuring a transparent verification process recorded on the blockchain. Incentives for participation and decentralized governance enhance accountability. Integration with existing platforms streamlines verification across social media and news aggregators. Through decentralization and community-driven verification, our DApp strives to provide a reliable solution to combat misinformation, restoring trust in digital news dissemination.

3. Timeline

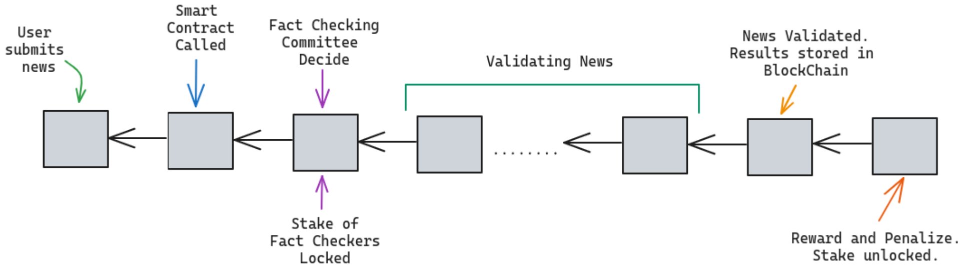

User submits news
Smart Contract Called
Fact Checking Committee Decide
Validating News
News Validated. Results stored in BlockChain
Stake of Fact Checkers Locked
Reward and Penalize. Stake unlocked.

Figure 1: Timeline of the News Validation

4. Issues Handled in the Smart Contract

1. **Sybil Attack**

   The smart contract needs a stake to be deposited to register a validator as a fact checker. This stake will be locked for the entire period of time this news gets validated. Part of this stake will be deducted if a validator votes wrongly on a news and the rest will be refunded back to the validator.

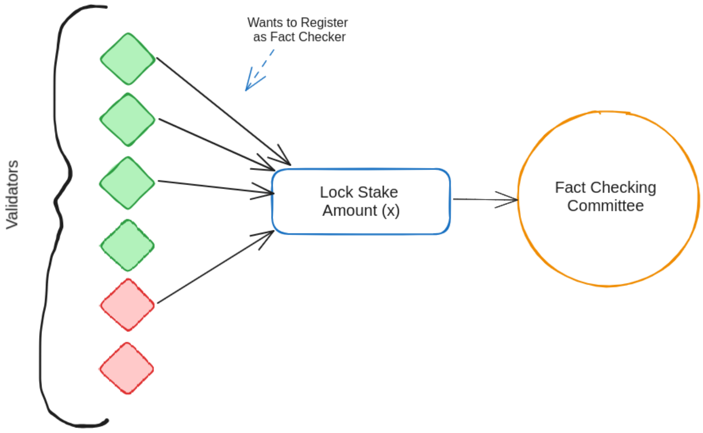


Figure 2: Locking Stake to become Fact Checker

This method called ”Staking” will make the system Sybil resistant by making it expensive for attackers to gain control over the network. If a validator acts maliciously, their staked coins are destroyed, making deceit a costly endeavour.

2. **Weight of Votes**

   Each fact checker will vote in either positive or negative. Positive vote means that the fact checker finds the news as legit and negative vote means they find the news as fake. The vote range is (0,2] i.e. lies from 0 to 2, with 2 included.

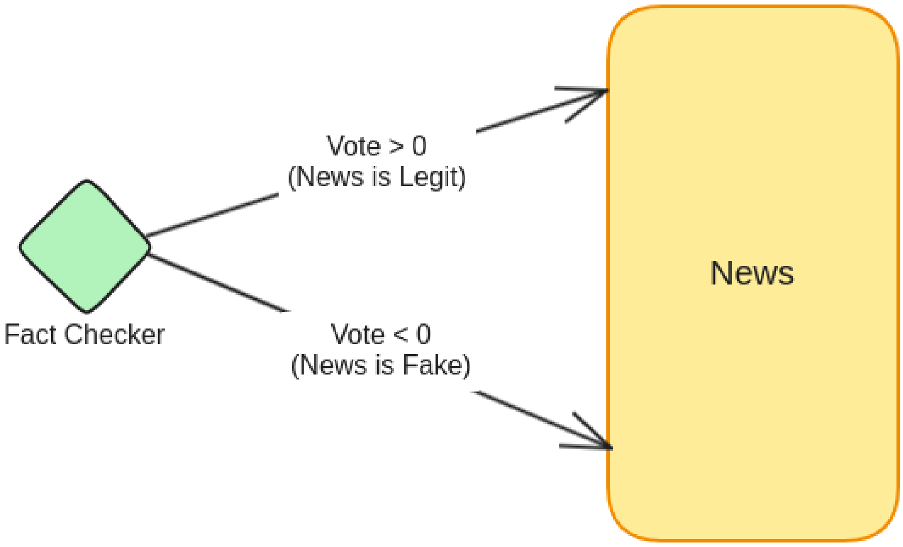


Figure 3: Fact Checker either votes in positive or negative

Each fact checker has a default rating and expertise category at the time of bootstrapping. The initial vote starts at 1, and then increases or decreases based on the current rating of the validator. If a validator acts maliciously by false voting, its rating will decrease which will give them less vote weight. If a validator acts honestly by correct voting, its rating will increase which will give them more vote weight.

```
vote=rating/DEFAULT_RATING
if (news.category==validator.expertise_category) {
    vote=2*vote
}
```

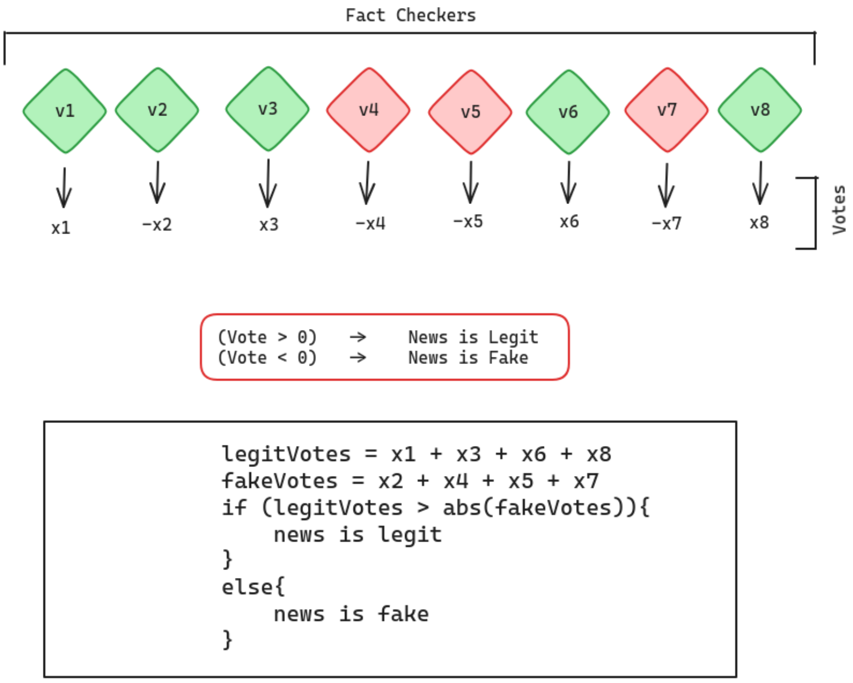
If a validator has expertise in the news category it is validating, then its vote will be multiplied by a factor of x ($= 2$), giving the vote more weight.

Now if a malicious validator, acts honestly to increase the rating, then starts behaving maliciously after several rounds, it can cast only a maximum vote of 2. It will not get support from the rest of the malicious nodes (if any) as their rating will go down to a minimum after those rounds, while the rating of honest validators will be much higher than those of malicious validators, supporting the correct votes with higher votes. Thus, this system is resistant to adaptive adversaries.

3. **Methods to Evaluate**

   Each validator's vote will be added to make summation votes based on the favour of the news they have chosen. The positive votes will be added together to get the total votes in favour that the news is legit.

The negative votes will be added together to get the total votes in favour that the news is fake. The summation votes with higher absolute values will be chosen as the validated result whether the news is legit or fake.



```
legitVotes = x1 + x3 + x6 + x8
fakeVotes = x2 + x4 + x5 + x7
if (legitVotes > abs(fakeVotes)){
    news is legit
}
else{
    news is fake
}
```

Figure 4: Evaluating based on Majority Votes

4. **Rewarding and Penalize Voters**

   Voters who vote the same as the majority voters will be considered **True Voters** and rewarded based on that, while the voters who didn't vote the same as the majority voters will be considered as **False Voters** and penalized based on that.

   A part of the stake locked will be penalized for each of the false voters, while the rest will be refunded. The rating for them will also decrease by a factor.

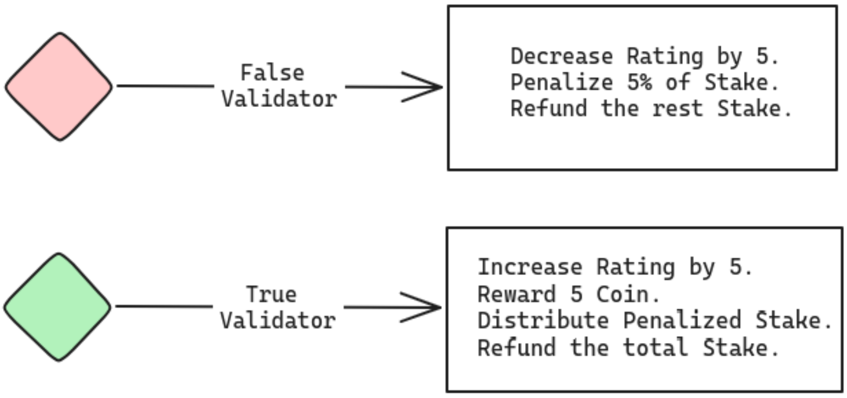


Figure 5: Penalty and Reward for Validators

The stake penalized from the false voters will be distributed among the true voters along with a reward fee, which is given by default to every true voter.

5. **News Item Upload** We can use a hash of the news content (similar to the newsHash) to uniquely identify news items. The hash can be generated off-chain and submitted along with the news content.

```
// Structure to represent a news item
struct NewsItem {
    // Address of the user who uploaded the news item
    address uploader;
    // Title of the news item
    string title;
    // Content of the news item
    string content;
    // Timestamp when the news item was uploaded
    uint256 timestamp;
    // Flag to indicate if the news item has been verified
    bool verified;
}

// Mapping to store news items
mapping(uint256 => NewsItem) public newsItems;

// Event to log when a news item is uploaded
event NewsUploaded(uint256 indexed itemId,
address indexed uploader, string title, uint256 timestamp);

// Function to upload a news item
function uploadNews(string memory _title,
string memory _content) public {
    // Generate a unique ID for the news item
    uint256 itemId = uint256(keccak256(abi.encodePacked(
    msg.sender, block.timestamp)));
    // Store the news item
    newsItems[itemId] = NewsItem(msg.sender, _title,
    _content, block.timestamp, false);
    // Emit an event
    emit NewsUploaded(itemId, msg.sender,
    _title, block.timestamp);
}
```

NewsItem Struct: This struct represents a news item and contains fields such as the uploader's address, title, content, timestamp, and a flag indicating whether the news item has been verified.

newsItems Mapping: This mapping is used to store the news items uploaded by users. The key is a unique identifier for each news item (e.g., a generated ID), and the value is a NewsItem struct.

uploadNews Function: This function allows users to upload news items by providing a title and content. Inside the function, a unique ID for the news item is generated using the uploader's address and the current timestamp. The news item is then stored in the newsItems mapping, and an event (NewsUploaded) is emitted to log the upload event.

Users can call the uploadNews function to submit news items to the DApp, which can then be evaluated and verified by other users or mechanisms within the DApp.

6. **Bootstrapping**

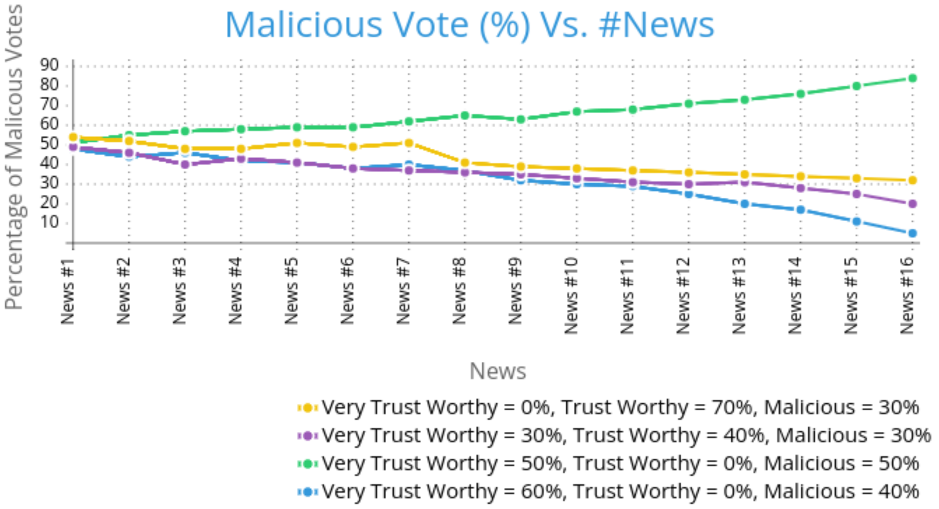
   Each validator will start validating with an ideal default rating, which will eventually increase or decrease based on the vote it is casting. Trustworthiness increases with the increase of correct validations.

   So, the system will bootstrap with a default rating and an expertise category for each validator.

5. Simulation

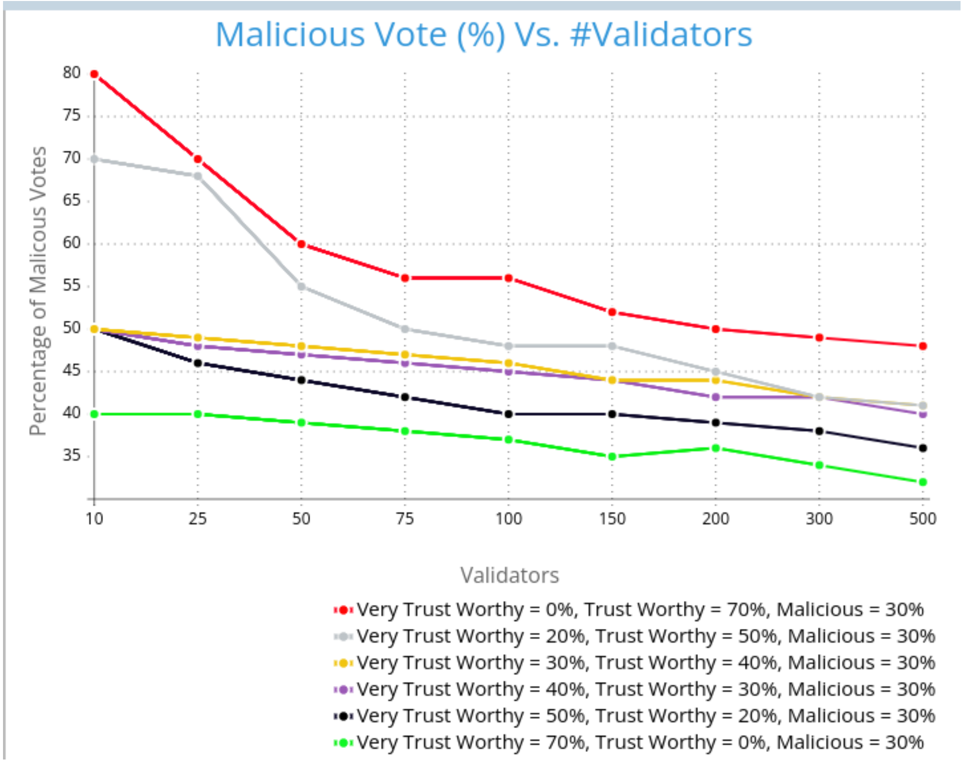
5.0.1 Malicious votes get less weight over time

We implemented a voter system with varied the percentages of Very Trust Worthy, Trust Worthy and malicious voters. We feed the system with x news and track the percentage of Malicious Votes generated. It is visible, the percentage of Malicious Vote will get closer to 0 over time, each time a news is correctly validated.



5.0.2 Fact Check with 30% Malicious Nodes

We vary the number of nodes from 10 to 500, the percentage of Trustworthy and Very Trustworthy to get the Malicous Vote (%) for 1 news. With the increase in number of nodes, the resistance to malicious takeover increases. This is due to the increasing total weight of the valid votes compared to the malicious vote weights.


Malicious Vote (%) Vs. #Validators
Percentage of Malicous Votes
Validators
Very Trust Worthy = 0%, Trust Worthy = 70%, Malicious = 30%
Very Trust Worthy = 20%, Trust Worthy = 50%, Malicious = 30%
Very Trust Worthy = 30%, Trust Worthy = 40%, Malicious = 30%
Very Trust Worthy = 40%, Trust Worthy = 30%, Malicious = 30%
Very Trust Worthy = 50%, Trust Worthy = 20%, Malicious = 30%
Very Trust Worthy = 70%, Trust Worthy = 0%, Malicious = 30%