A Blockchain Based Reputation System

Saad Sher Alam, Shazer Ali, Imaan Hameed, Jaffer Iqbal

August 2022

1 Overview

We present a reputation system atop the Ethereum blockchain, with a novel reputation quantification protocol, and robust security guarantees. To the best of our knowledge, this is the first reputation system that addresses collusion attacks, unfair rating problem, sybil attacks, and re-entrance attacks effectively.

The system comprises of 3 important components: The Ethereum blockchain and smart contracts, a distributed file-sharing system, and a novel reputation quantification protocol.

The Ethereum blockchain serves as a decentralized and a transparent source to store seller ratings. The decentralization eradicates any trust issues and the transparency allows the public to verify changes and updates. The smart contracts enable the system to register new sellers, make transactions between buyers and sellers, and update the reputation scores of sellers. The distributed file-sharing system is used as a more efficient storage mechanism (compared to a blockchain) for product details such as product ID, pictures, reviews, comments, and other information for every seller. The distributed file-sharing system also maintains a file, SellerIDList, that contains a list of all active sellers in the system. Lastly, the novel reputation quantification protocol is designed to mitigate the threats of collusion and unfairness in reviews.

The article is structured as follows: Section 2 establishes important terminology; Section 3 goes in the depth of architecture and the phases associated with the system; Section 4 covers the details of the reputation quantification protocol; Section 5 highlights current limitations and future work.

2 Terminology

Before we dive into the details of the architecture and the reputation protocol, it is important to establish some terminology to avoid any confusions in the upcoming sections.

Reputation: A reputation score, in the range $0-1$, calculated in a way that mitigates the impact of collusion and unfair ratings on the reputation.

Review: A review left by the buyer for a product, consisting of two parts: rating and comment.

Rating: A discrete numerical score, ranging from $1-3$, to represent product quality. By definition, a score of 1 means a negative rating, a score of 2 means a neutral rating, and a score of 3 means a positive rating.

Comment: A descriptive comment based on product quality and buyer experience.

Transaction: A purchase of a product by a buyer from a seller. A review is left after every transaction.

Weight (W): A weight, in the range $0-1$, assigned to a review based on the likelihood of collusion and unfairness for the review.

Transaction Score/Impact (I): A numerical value for the current transaction between a buyer and a seller based on the W and rating, that determines the impact of that transaction on the final reputation of the buyer.

$\mathbf{F_{Score}}$**:** The final reputation score for a seller based on k previous reviews.

SellerPubKey: Public key of the seller.

BuyerPubKey: Public key of the buyer.

IPFS: Inter Planetary File System, a distributed file sharing service used to store SellerIDList and SellerProductDetails.

SellerIDList: A file containing a list of all verified sellers on the system. The SellerIDList is stored on the IPFS.

SellerProductDetails: A file containing product details such as product ID, product images, and previous reviews. SellerProductDetails is stored on the IPFS.

We strictly adhere to this terminology. In case of any ambiguity in the upcoming sections, please refer to this section.

3 The Architecture

3.1 Registration Phase

Every new seller is required to register themselves through the **RegisterSeller** smart contract. The registration phase requires interaction with a distributed file-sharing service, for example IPFS. All seller IDs, product details, and SellerIDList will be stored on the IPFS. For every new seller, **RegisterContract** checks if that seller is already a part of the system by fetching SellerIDList from IPFS. If not, RegisterContract appends the seller ID to the SellerIDList, stores SellerProductDetails, and returns the address of the file. This address (location where SellerProductDetails is stored on the IPFS), is stored on the blockchain with each seller's reputation score and SellerPubKey.

Algorithm 1: RegsiterSeller

```
   Input  : SellerID, SellerProductDetails, SellerListAddress
   Output: SellerAddress
1  SellerList ← FETCH(SellerListAddress);
2  if SellerID not in SellerList then
3      WRITE(SellerList, SellerID)
       SellerAddress ← STORE(SellerID, SellerProductDetails)
       return SellerAddress
4  end
```

By the end of registration phase, every seller will have their product details stored on the IPFS, and the seller ID stored in the SellerIDList. Their reputation will be initialized to 0. The seller can now move on to the 'No Trust Phase.'

3.2 No Trust Phase

For a new seller, the system can not assume any reputation score. In other words, for a seller with no prior reviews, it is uncertain if the seller is trustworthy or not. To make the reputation system more robust to re-entrance attacks from malicious sellers, we propose a 'No Trust Phase' governed by the **NoTrustContract**.

For the first k transactions for every new seller, the seller will have to stake ETH equivalent to the product price. Upon successful execution of the **TransactionContract**, the ETH is returned to the seller. If the transaction is not completed faithfully, the ETH staked is given as a refund to the buyer. This is based on the assumption that any honest seller will have no problem in staking ETH till they build up an honest reputation. The buyer will leave reviews for these k transactions. However, the seller's reputation won't be effected by these reviews, i.e for k transactions the reputation will stay 0. As soon as the k+1'st transaction is completed successfully, the seller will receive their first reputation score. This reputation score will be published on the blockchain along with

SellerPubKey and the address of SellerProductDetails by the **UpdateReputationContract**.

Note that 'No Trust Phase' limits the system from having a very large value for k. In the case where a seller sells expensive products, it is not feasible for the seller to wait for, say 10000 transactions, and stake equivalent ETH for that amount of transactions to receive a reputation score. Once the seller receives their first reputation score, the seller can proceed to the 'Regular Operataion Phase.'

Algorithm 2: NoTrustContract

```
  Input  : SellerPubKey, ProductPrice, BuyerPubKey
1 StakedETH ← WITHDRAW(SellerPubKey);
2 CALL(TransactionContract);
3 if TransactionContract successfull then
4     TRANSFER(StakedETH, SellerPubKey)
5 else
6     TRANSFER(StakedETH, BuyerPubKey)
7 end
```

3.3 Regular Operation Phase

The regular operation phase commences for a seller who has received a reputation score and has more than k reviews. For every transaction, a buyer and a seller execute the **TransactionContract**, which transfers ETH from the buyer to the seller. The buyer also has to pay a small review fee, which will be refunded once the buyer leaves a review. This is to ensure that everyone leaves a review. Once the buyer leaves a review, **UpdateReputationContract** uses the reputation quantification mechanism described in section 4 to update the reputation for the seller, and publish it on the blockchain alongside the SellerPubKey and address of SellerProductDetails. The details for the **TransactionContract** are descirbed below:

Algorithm 3: TransactionContract

```
Input   : SellerPubKey, ProductPrice, BuyerPubKey, ReviewFee
1 Payment ← WITHDRAW(ProductPrice, BuyerPubKey);
2 ReviewFee ← WITHDRAW(ReviewFee, BuyerPubKey);
3 Deliver(Product);
4 TRANSFER(Payment, SellerPubKey);
5 Review ← GetReview(BuyerPubKey);
6 if Review not EMPTY then
7     TRANSFER(ReviewFee, BuyerPubKey)
       Reputation ← CALL(UpdateReputationContract);
8     PUBLISH(SellerPubKey, Reputation);
9 end
```

4 A Novel Reputation Quantification Protocol

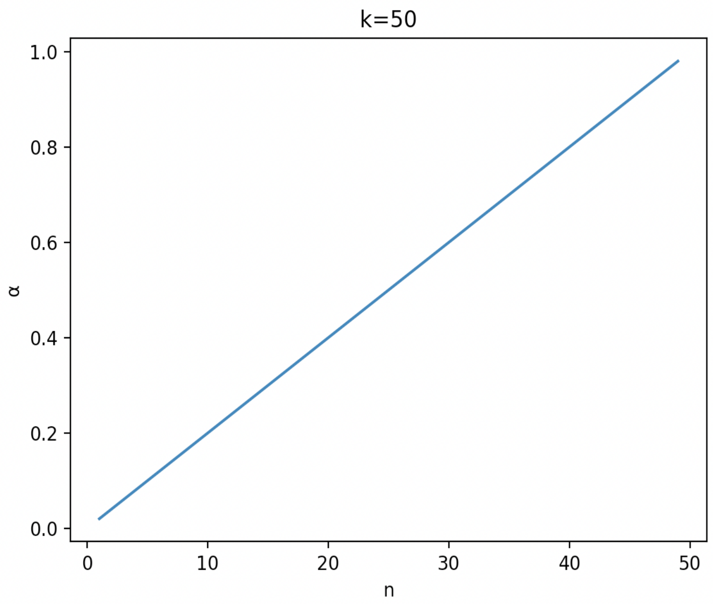
We propose a novel reputation quantification protocol, which to the best of our knowledge, is the first ever that provides robust guarantees against collusion attacks and unfair rating attacks. We introduce a "weight" for the reputation scores for each transaction between a buyer and a seller, which is used to calculate the transaction score for the current transaction based on the likelihood of collusion, an unfair rating, or both. The proposed protocol is also cross-platform, i.e. the reputation scheme is not limited to e-commerce applications but can be used in any reputation based setting. The section is structured in the following way: 4.1 The Unfair Rating Problem, 4.2 Collusion Attacks, 4.3 The Final Review Weight, 4.4 The Big Picture: Final Reputation.

4.1 The Unfair Rating Problem

Centralized reputation systems suffer from a flaw: the buyer enjoys the freedom of leaving any review, irrespective of if the review is the true representation of the quality of the product. For genuine sellers, this results in an unfair degradation of reputation. It is impossible to make the buyer always write an honest review, however it is possible to adjust transaction score based on if the review is fair or unfair. We try to predict unfairness for every review using an unfairness parameter α. α is defined as follows:

$$\alpha = \frac{n}{k}$$

k is a system design choice, defined as the number of previous reviews to be considered, and n is the number of previous reviews with the same rating (1,2, or 3) as the current review. In the case where the review is unfair, n $\rightarrow$ 0. Therefore, $\alpha \rightarrow 0$. In the case where the review is fair, it matches the previous trend of reviews for the seller, i.e. n $\rightarrow$ k and $\alpha \rightarrow 1$. Therefore, $\alpha \in [0, 1]$. The visualization, for $k = 50$ is shown below:



Now, consider the case of a review that is the first to be identified as classified differently from the previous reviews, such as the first negative review after k-1 positive reviews. In this case, n=0 so $\alpha = 0$. Assuming that the k^{th} review was not unfair, it still received an α rating of 0. However, this is not counter-intuitive as α does not render the review useless. As the product continues to receive negative reviews, n will rise over-time, leading to an increase in α, to the point that further negative reviews will not be considered unfair - they will have a higher α value. This allows α to take a trend-based approach that catches unfairness while removing fluctuations in the reputation score, allowing for a more stable picture of the reputation score for the product.

In short, we use α as a measure of unfairness, the lower the value of α, the more chance of the review to be unfair and vice versa.

4.2 Collusion Attacks

Collusion attacks are predetermined agreements between two parties in the system to collaborate to defraud the system. An example scenario can be an agreement between a seller and a buyer, where the buyer makes a large amount of transactions to leave positive reviews for the seller to increase reputation. There are two important indicators of collusion in a reputation system. Firstly, the number of transactions between the same buyer and a seller. To successfully defraud the system, the buyer and the seller will have to make a large amount of transactions between them such that there can be a significant effect on the reputation score. Secondly, the product price. The more expensive the product, the more infeasible it is to buy it in bulk to leave reviews. Our protocol represents these two indicators as β and γ respectively.

$$\beta = \frac{k}{n}$$

$$\gamma = \frac{Price - Price_{min}}{Price_{max} - Price_{min}} \tag{1}$$

k is defined as the previous number of reviews to be considered (the exact same as α), and n is the number of previous reviews left by the same buyer (current review inclusive). As n $\rightarrow$ k, $\beta \rightarrow 1$ and as n $\rightarrow$ 1, $\beta \rightarrow k$. Thus, $\beta \in [1, k]$. This means, for a seller with a higher number of transactions with the same buyer, β takes a lower value. γ is the price, which is normalized using $Price_{min}$ and $Price_{max}$, which is a design choice for the system. Hence, $\gamma \in [0, 1]$. For the sake of consistency and simplicity, β is also normalized. The normalization makes sure that $\beta \in [0, 1]$.

$$B_{norm} = \frac{k - n}{n(k - 1)} \tag{2}$$

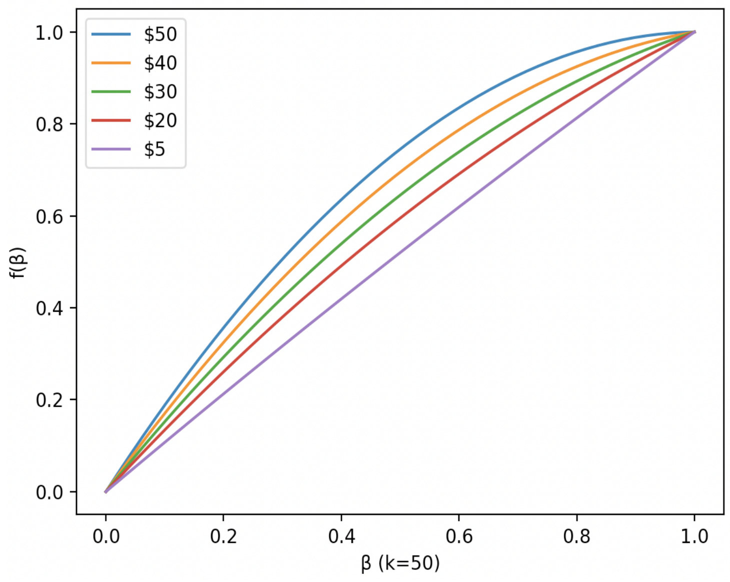
Now that we have established the two parameters that impact collusion, it is important to understand how they interact with each other to determine the likelihood of collusion. Suppose the case where $\beta \rightarrow 1$. This means that there is 0 previous transactions between the seller and the buyer, not including the current transaction i.e. the current transaction is the first transaction between a buyer and seller pair. Since it's the first transaction, there's no indication for collusion. Hence, γ (or the price) of the product should not impact the reputation score of the seller. Now, consider the case where $\beta \rightarrow 0$. This is the case where every transaction for a seller is with the same buyer as the current transaction. This is a strong indication of collusion. For the cases where β is neither 1 nor 0, the collusion likelihood depends on γ. The more expensive the product, the less likely the collusion. Let's try to represent this mathematically.

$$f(\beta) = \begin{cases} 1 & \beta \rightarrow 1 \\ \beta\gamma & \beta \rightarrow 0 \end{cases}$$

Interpolating, we get:

$$f(\beta) = \beta + (1 - \beta)\beta\gamma \tag{3}$$

Let's try to visualize this:



As it can be seen from the graph, for the case where $\beta \to 1,\ f(\beta) \to 1$. For high values of β the impact of γ on $f(\beta)$ is negligible. When $\beta \to 0,\ f(\beta) \to 0$. For β values in the middle, $f(\beta)$ depends on γ for products of different prices. Conclusively, $f(\beta)$ integrates β and γ to determine the likelihood of collusion. $f(\beta) \in [0, 1]$. As $f(\beta) \to 0$, the chance that the current review is colluded is higher. We use $f(\beta)$ as a measure for collusion.

4.3 The Final Review Weight

In sections 4.1 and 4.2 we established α and $f(\beta)$ as measures for unfair rating and collusion respectively. To recap, $\alpha \to 0$ as $n \to 0$ i.e. the review is unfair and $f(\beta) \to 0$ as $\beta \to 0$ i.e. the review is a result of collusion. Also recall that $\alpha \in [0, 1]$ and $f(\beta) \in [0, 1]$. For an ideal reputation system, we need to give a lower weight to a review if it's either unfair, or a collusion, or both. Therefore, we define the weight, W, of a review as a harmonic mean of α and $f(\beta)$.

$$W = \frac{2\alpha f(\beta)}{\alpha + f(\beta)} \tag{4}$$

Note that $W \in [0, 1]$. Based on experimental analysis, we can further refine (4) using a weighted harmonic mean:

$$W = \frac{2(w_1 + w_2)\alpha f(\beta)}{w_1\alpha + w_2 f(\beta)}$$

where w_1 and w_2 are arbitrary constants.

4.4 The Big Picture: Final Reputation

To find the transaction score of the current transaction for a seller, we will require two parameters: transaction weight, and the rating provided by the buyer on the scale of [1,2,3]. We will calculate transaction score **I** by simply taking **product** of the rating and weight. The weight will essentially act as a truth value for the transaction. Hence, more weight will result in a higher transaction score. Impact of transaction **i** is defined as follows:

$$I_i = \ Rating_i \times Weight_i$$

The crux of finding the final reputation score is to simply take the ratio of the summation of all impacts with the summation of all the maximum possible impacts. A maximum possible impact $\mathbf{Max_i}$ is simply the product of the maximum rating, i.e. 3, and the actual impact of the transaction.

$$Max_i = \ 3 \times Weight_i$$

For the **k** transactions, the final reputation score will be:

$$F_{Score} = \frac{\sum_{i=1}^{k} I_i}{\sum_{i=1}^{k} 3 \times Weight_i} \tag{5}$$

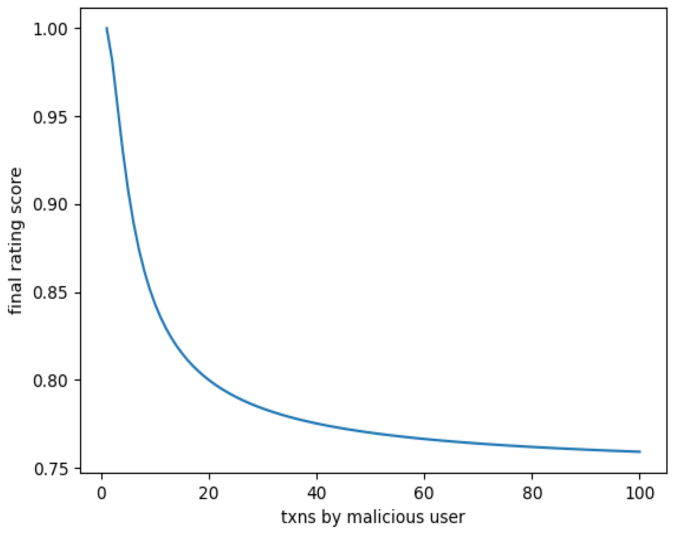
Since , we are considering rating values of [1,2,3] , F_{Score} will range between [$\frac{1}{3}$, 1]. Hence we need to normalise the F_{Score} such that values of our final score span in [0 , 1]

$$F_{score} = \frac{F_{Score} - \frac{1}{3}}{1 - \frac{1}{3}}$$

4.5 Properties of Final Reputation score

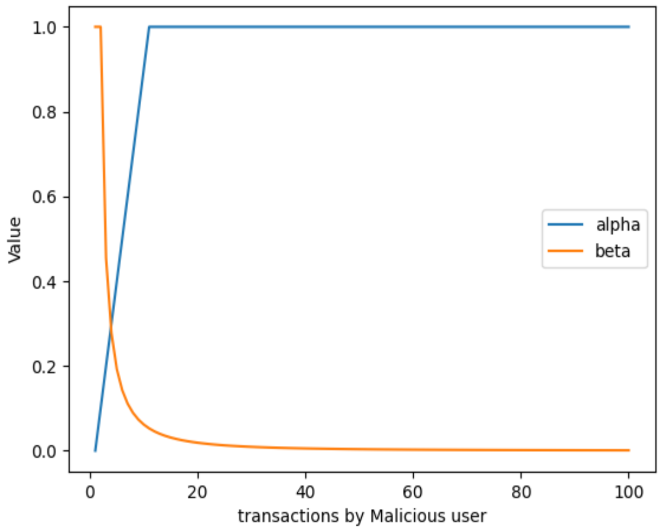
Using the ratio outlined above, we will achieve a normalised score between 0 and 1, where $F_{score} \to 1$ indicates higher ratings and $F_{score} \to 0$ lower ratings for k previous transactions.

Incorporating weights in our F_{score} reduces the danger of the reputation of sellers being affected by malicious parties. To demonstrate the resistance of our model to such attacks, we present a simple scenario where an honest **Seller A** has received a rating of 3 from 10 successful transactions made by 10 honest buyers. Hence for these 10 transactions, $F_{score} = \frac{30}{30}$. In order to slash the reputation of **Seller A**, a malicious user has to maka a transaction and leave a minimum rating. Below we present the graph to show the extent of the impact the malicious user will have in lowering the F_{score} by generating up to **100 transactions**:



In the figure above we observe that despite 100 transactions, each having a rating of 0, the F_{score} is still > 0.75. In fact, even if a larger number of transactions were allowed, the decrease in F_{score} would not have been significant as our graph illustrates that the magnitude of the gradient is decreasing in an exponential manner which is a direct indication of very low weightage being assigned to these transactions. We also notice that the most decrease in F_{score} occurred till the 20^{th} transaction which caters to real life scenario where an honest buyer may buy the same commodity multiple times and thus, impact the F_{score}.

The shape of our graph of F_{score} is directly linked to the behavior of our fundamental parameters, α and $f(\beta)$, which will essentially determine the role of W used in the **Impact** I of a transaction. Recall that ourα increases as more transactions have the same rating while our β decreases as the number of transactions with the same buyer increases. For the case above, we observe that with more transactions by a malicious user, α will increase α and β will decrease. In the graph below, we present the variations in these two variables with increasing transactions made by the said malicious user:



From the graph, we come to learn that despite the increase in value of α, β has decreased significantly and by the time α reaches its maximum value of 1, β has already approached 0. Recall that to find W of a transaction, we use **harmonic mean** and since our β is already close to 0, W will tend to have a lower value regardless of α reaching its maximum. Thus, the mathematical model is successful in resisting the attacks with the intention of varying F_{score} in the case of a single adversary.

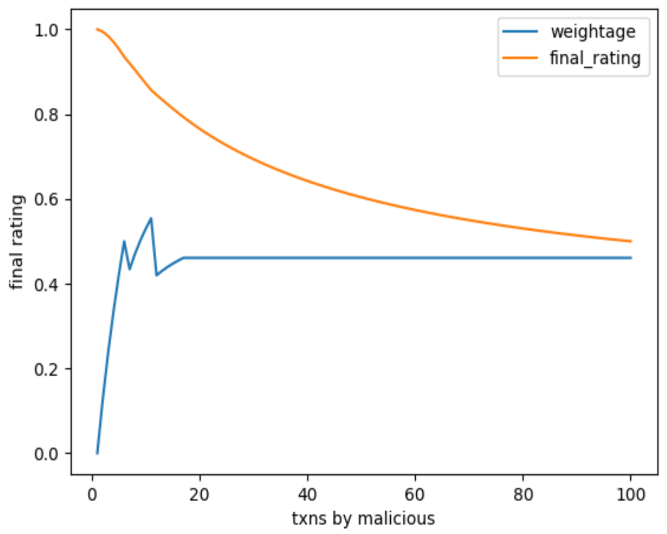
It is important to note that this does not mean that our reputation system will not penalize sellers for negative reviews. This is the case where multiple negative reviews are left by the same buyer, which is an indication of adversarial manipulation. In this case, β for the transactions of this buyer approaches 0, hence the weight for the reviews is low. However, in the case where the multiple buyers leave negative reviews for a seller, that will be an indication for poor product quality, β for those reviews will be high, and they will have a high weightage and a greater impact on the final reputation.

5 Limitations and Future Work

Our robust quantification model presents great resistance against collusion attacks where an adversary attempts to either increase or decrease rating of a seller. However, our resistance is limited against collusion where multiple parties are involved in unfairly changing rating of seller/s.

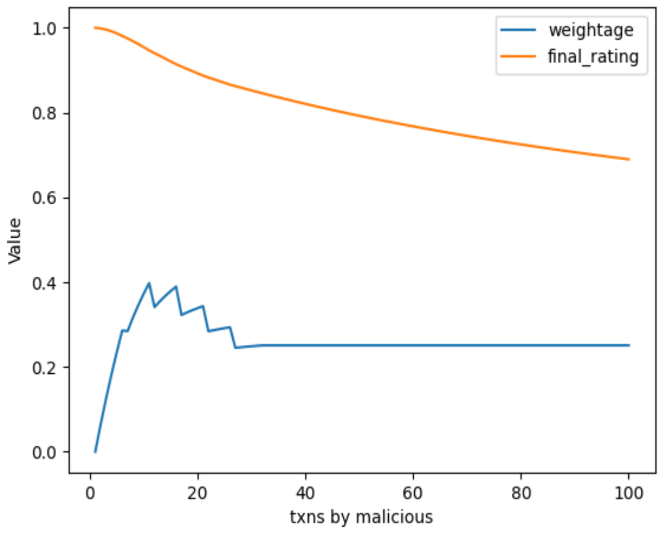
The limitation stems mainly from the incapability of window size, k, to capture repeated transactions by same user in unfairly rating the seller. As explained before, in the case of an adversary, increasing number of adversarial transactions causes the window of k previous transaction to be filled up by adversarial reviews, which leads to $\alpha \to 1$. In this case of a single adversary, the reputation score was sustained because repeated transactions from one adversary can be caught by β. However, in the case of multiple adversaries colluding together to provide same rating to seller on multiple transactions, the drop in β can easily be reduced which will lead to a higher weightage and transaction score for an adversarial review. An even more serious threat will occur once the number of adversaries will become equal or to or greater than k, in that case there is no barrier from stopping each adversary transaction to gain maximum weight i.e causing drastic change in the F_{score}.

To show the effect of a such case, we provide you with a scenario where we have 5 malicious users to lower the F_{score} of an honest seller. In case 1, we set the window size to 15 , meaning that each user will be buying the product thrice (so that $\alpha \to 1$) to create most effect on overall score. The user will buy the product according to their turn set as their public key address in ascending order and cycle will continue till 100 transactions. The first k transactions will be filled by honest users, each giving max rating to the seller.



In the figure above, the zig-zag, sharp peaks, are result of buying pattern for previous k transactions and does not hold much relevance to the problem. Significance lies in the fact, that after around 15 transactions, the W has become constant all the way up to 100 transactions. This is due to fact that after 15 transactions by all the users combined, the values of α and β have become constant. Given constant W, malicious users can change the F_{score} to any direction by just buying the product at more frequency than honest buyers.

In case 2 we have kept everything constant except k, which is set to 30.



In case 2, we observe that, the value of β at which constant weight occurs is lower than that in case 1 only because, now each malicious user has to buy 6 items rather than 3 times causing lower value of beta at which it becomes constant. Hence, we observe that by the end of 100 transactions the drop in F_{score} is comparatively less than in Case 1.

Increasing k can reduce effects of adversary but does not completely mitigate the threat. Additionally, it is important to note that a considerable value of k will also result in a very long No Trust Phase for new sellers. Hence, our focus will now be in creating changes to our model in effort to eliminate the threat of multi-party collusion attack to a greater extent.

Alongside the security problem explained above, we have identified the following areas that require future work. Firstly, we need to come up with a more thorough Sybil attack solution which requires more research on KYC. We will have to update our registration phase based on the solution for Sybil attack. Secondly, we require more exploration on distributed file sharing systems to motivate the use of IPFS. We plan to include a comparison between popular distributed file sharing systems to justify the use of IPFS (or any other system) for our application.