BLOCKCHAIN-BASED TRUST MODEL: ALLEVIATING THE THREAT OF MALICIOUS CYBER-ATTACKS

A Dissertation
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Ahmed Youssef Bugalwi

In Partial Fulfillment of the Requirements
for the Degree of
DOCTOR OF PHILOSOPHY

Major Program:
Software Engineering

April 2020

Fargo, North Dakota

North Dakota State University
Graduate School

Title

BLOCKCHAIN-BASED TRUST MODEL: ALLEVIATING THE THREAT OF MALICIOUS CYBER-ATTACKS

By

Ahmed Youssef Bugalwi

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

DOCTOR OF PHILOSOPHY

SUPERVISORY COMMITTEE:

Dr. Kendall E. Nygard
Chair

Dr. Oksana Myronovych

Dr. Pratap Kotala

Dr. Fred Riggins

Approved:

April 21, 2020	Dr. Kendall E. Nygard
Date	Department Chair

ABSTRACT

Online communities provide a unique environment where interactions performed among its subscribers who have shared interest. Members of these virtual communities are typically classified as trustworthy and untrustworthy. Trust and reputation became indispensable properties due to the rapid growth of uncertainty and risk. This risk is a result of cyber-attacks carried out by untrustworthy actors. A malicious attack may produce misleading information making the community unreliable. Trust mechanism is a substantial instrument for empowering safe functioning within a community. Most virtual communities are centralized, which implies that they own, manage, and control trust information without given permission from the legitimate owner. The problem of ownership arises as actors may lose their reputations if the community decided to shut down its business. Sharing information is another valuable feature that aids lessening the impact of dishonest behavior.

A new trust model called "*TrustMe*" was developed in this research as a reliable mechanism that generates precise trust information for virtual communities. TrustMe consists of several factors that aim to confuse untrustworthy actors, and to make the generated trust score is hardly reversed. A blockchain-based trust model is also developed to address the problem of ownership as well as offering a decentralized information sharing mechanism through a distributed application called "*DATTC*." The efficiency of the proposed models was identified by conducting various analytic experimental studies. An unsupervised machine learning method (density-based clustering) was applied using two different datasets. Also, graph analysis was conducted to study the evolvement of communities and trust by finding connections between graph metrics and trust scores generated by TrustMe. Finally, a set of simulations using stochastic models to evaluate the accuracy and success rates of TrustMe, and a simulation set mimicked the blockchain-model in alleviating the influence of Sybil attack. The relationships among actors were hypothesized as actors divided into trustworthy and untrustworthy performing cooperative and malicious attacks. The results of the study prove that TrustMe can be promising and support the first hypothesis as TrustMe outperformed other trust models. Additionally, the results confirm that the blockchain-based trust model efficiently mitigates malicious cyber-attack by employing cross-community trust and preserves ownership property.

ACKNOWLEDGMENTS

This dissertation is deemed evidence of a long journey of continuous study, effort, and research. During this journey, substantial assistance has been offered to make this work feasible. Thus, I would like to express my sincere gratitude to those who gave such help and support.

First and foremost, I would like to thank and express my appreciation to my supervisor, Professor Kendall E. Nygard. I am primarily indebted to Dr. Nygard for his exemplary guidance, invaluable advice, knowledge, encouragement, and endorsement, which make this work a triumph. I would further thank Dr. Fred Riggins, Dr. Oksana Myronovych, and Dr. Pratap Kotala for serving on my committee. Their valuable expertise, comments, and precious time were of vital importance to the success of my research. Also, I would like to thank North Dakota State University and particularly the Computer Science Department for providing assistantship while completing my study.

Exceptional gratefulness, further, goes to my aunt Dr. Kheria Bugalwi and her husband, Dr. Abdulsalam Elmesallati, who both have always been a source of support and encouragement throughout my Ph.D. journey. I am also extremely thankful to my lovely family, darling mother, brothers, sisters (specifically Amal Bugalwi), nephews, nieces, brothers-in-law, sisters-in-law, relatives, my wife's family, and friends for their backing and assistance.

Last but not least, exceptional thanks go to a unique person who is the beginning and end of this story. I would say that without you, this success would not have been possible. My wife, Nagla, we are partners in this success; it is yours as mine. Thank you for your patience, support, inspiration, and love throughout this long journey.

DEDICATION

This doctoral dissertation is dedicated in honor of the soul of my beloved father (*Youssef M. Bugalwi*). Your words and wise still guide and hearten me throughout life.

"*You will never be forgotten.*"

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

1.1. Overview

The rapid growth of new technologies introduces many solutions to a range of today's problems but, at the same time, comes with several challenges. The world of the internet is evolving with new emerged technologies such as cloud computing and blockchain. These technologies are designed to leverage the use of online applications such as open electronic markets, peer-to-peer (P2P) applications, social media. These applications foster the interaction between users; people can easily interact with each other to use services, to sell/buy products, to chat, to deliver/receive goods, etc. Online applications involve entities and individuals as central players. As these entities play a vital role in online communities, their actions have a significant influence on the system's reliability and the experiences of others.

P2P communities are distributed decentralized applications that directly allow peers to transact, share, and exchange information [1]. The dynamism of P2P communities means that nodes/peers can join and leave without informing the others. In this case, anonymity, risk, and uncertainty become key attributes of such communities. Authors [2] state that the rate occurrence of cyber-attacks is increasing, meanwhile it enforces entities to find out sufficient protection. Potential incomplete or distorted information can be exploited by malicious attacks that urge the need to establish a mechanism that aids in addressing and managing risks and uncertainty.

Generally, online users need to make their own decisions and judge others with the absence of face-to-face cues [3, 4]. With the lack of knowledge plus the lack of historical information, how can people trust each other? The rate of risk usually reaches at high levels when dealing with anonymous entities/individuals. Therefore, addressing the issue of uncertainty is to develop a mechanism that can establish trust and reputation relationships among peers, and the trust level or trust score should be accessible to all members in a community. Trust has become an essential dimension that helps members achieving their reasonable expectations.

Although the development of trust mechanism can assist online entities, the majority of the proposed models/mechanism is still subjected to malicious behaviors which means trust can be manipulated. Many researchers have proposed a large set of reputation-based trust models to address uncertainty problems and ease the decision-making process [5, 6, 7, 8]. However, most of these models deal with direct, straightforward malicious behaviors but fail to deal with clever malicious behaviors [9, 10]. Also, the miscalculation of the trust score is a major problem of computing reputations [11]. Several reputation systems measure the trustworthiness of users by summing a peer's ratings and finding the average to calculate the overall reputation score. Unfortunately, there is no way for such a system to distinguish between a peer who consistently receives a certain rating and a peer whose behavior has resulted in a variety of negative and positive reviews. Likewise, the number of reviewers is important in such a system, since, for example, a peer with a few good reviews in total can outshine a peer with thousands of mixed reviews.

Additionally, any data-driven model is customarily experienced the problem of cold-start. Trust models are among these data-driven models that encounter this problem. The difficulty of drawing a logical conclusion or drawing inferences for users or for whatever entity due to lack of sufficient information is called cold-start problem [12]. When a newcomer joins a community with no prior history or interaction, he/she will suffer from this problem [13, 14]. Cold-start is known as the main disadvantage/challenge of any reputation-based trust model. For example, when a new seller joins an online community and starts his/her own business, he/she has no prior transaction which means no history and no reputation [12]. Buyers may be afraid of dealing with sellers who have no reputation. The business owner will suffer at the beginning until building a good reputation, but the business may be gotten collapsed before building a reputation. According to the best knowledge of the researcher, there is no reputation-based trust model that addresses the problem of cold-start, all cold-start solutions are connected to recommender systems, not to reputation-based trust models. Therefore, a model that is called Trust-By-Contract is proposed as a complementary model to the TrustMe model to solve the cold-start problem.

Furthermore, the need for managing identity is particularly noticeable in the virtual world. Even in the real world, this problem still exists. One person may have a bunch of identity documents such as driver licenses, passport, library Id, and others. In cyberspace, online identity is globalized, and it does not recognize national boundaries. Thus, the problem is more complicated. Identity management (IdM) concerns the management of identities that are commonly connected to individuals, their authentications, authorizations, and privileges [15]. IdM models are confined to specific domains with a commitment to be tied to these domains' boundaries. The aim of these models is to find an environment that is secure and effective in reducing cost, time, and repetitive tasks. However, the main challenge is that the majority of these models are centralized. Being centralized means these models are controlled by a single entity and vulnerable to a data breach. The data breach has become a critical problem since it constantly happens on a regular basis. In 2017, Yahoo suffered a significant data leakage of its users' data which around 3 billion accounts were breached. Another accident occurred with Uber when 57 million users' private data got breached [16]. As soon as the leakage occurred, it is possible that the dark web gets advantages of masquerading identity, damaging reputation, and invading privacy.

This study is focusing on managing identity from the angle of trust and reputation. In many cases, it is sometimes unfair to have multiple identities and multiple reputations for the same person, particularly in e-commerce communities. For instance, one seller is running a business on Amazon; this seller may decide to expand the business and open a new account on eBay selling goods or services. The same seller has multiple accounts and multiple reputations. This seller may desire to migrate his/her good previous Amazon reputation to be used on eBay. All of these online platforms do have a mechanism to share reputation information. Here, blockchain technology can play a significant role in solving this problem. Building a blockchain-based trust model that can map all necessary information from different platforms to generate a trust score for each individual/entity. The idea of using blockchain brings decentralization which means all reputation data can be stored in a ledger that has no single authority, tamper-resistant, and transparency. Moreover, sharing information among multiple online platforms (cross-community trust) help to lessen the impact of malicious attacks that aim to manipulate reputation dishonestly. The foundation of this blockchain-based trust model is the TrustMe model that we propose in this study.

1.2. Statement of Problem

In peer to peer environment, a peer regularly deals with others to accomplish a payoff. However, collaboration in uncertain environments like cyberspace exposes peers to risk. Fraud and deception are common practices that grow continuously in cyberspace [2, 9, 10, 16]. Misplacing trust is a major element to be a victim of fraud due to two possible reasons: a lack of enough information or misleading information [17]. People need to identify with whom to deal and with whom not to deal. Classifying people into trusted and distrusted is not a simple process even with users who have a certain level of skills. This process is exhausted which is hard to be done manually. Before cooperation, peers need to investigate both the worth and risk of interacting with other peers. This helps peers to decide whether to cooperate or not. The evaluation process requires a trust model that serves as a decision criterion. Thus, delivering a precise trust score to serve as decision criteria is a critical step that assists users in making proper decisions easily and avoids interacting with malicious entities [6, 8, 7, 18].

Building a trust model is a big challenge. The trust model needs to generate a trust score for all participants and usually based on several factors. Several approaches and models have been suggested to generate trust scores and categorize users into trusted and distrusted. The majority of trust models employ rating values in calculating trust. However, these models usually ignore the possibility of dealing with a single rating between two peers. In other words, some rating systems use the principle of keeping only the last ratings. When a user provides a rating for a first transaction and provides a rating for a second transaction, the system will update the first rating by the second rating which means the first rating is disappeared.

Also, one of the main disadvantages of reputation-based trust models is their inability to address the problem of cold-start. Forming a newcomer's behavior and predicting his/her future actions is a major challenge for reputation-based trust models. This challenge is known as a cold-start issue that appears when newcomers boot for the first time. Assigning an accurate trust value for these newcomers requires a strong mechanism.

Moreover, one-identity but multiple evaluation issues. Here, blockchain plays a significant role. It has been stated that reputation-based trust models work better when getting a good volume of information that increases the accuracy of the generated trust score. Also, more information helps in detecting malicious attacks and malicious users.

1.3. Significance of the Research

The significance of the work is that it provides a reputation-based trust model that helps to reduce the uncertainty levels among users in a peer-to-peer environment. Malicious users may tend to deceive to change the reputation of one peer. Without such a trust model, those malicious users can achieve their goals easily. This model mainly aims to identify the trust score of a user through scanning the history of the user's transactions/interactions over time. The model can be applied by users themselves and online platforms as well. Before performing a transaction, participants can explore the trust score of each other by using this model as a tool to ease the process of decision-making. In this case, peers can lessen the likelihood of being victims of fraud.

Furthermore, online communities are vulnerable to scams and frauds which lead to the necessity of trust models. Users are highly subjected to malicious attacks or dealing with dishonest users. Thus, the proposed model is designed with a focus on reducing such malicious attacks. The impact of outlier ratings is minimized in this model. These outliers are usually a product of launching a malicious campaign in conjunction with temporal information to upgrade or degrade the reputation of the target.

Another important significance is that the proposed work targets to solve the problem of having multiple identities and multiple trust scores. Here, the importance of blockchain technology presents to be a possible solution to this problem by linking multiple separate online platforms to a decentralized platform that preserves all trust information. This station can provide trust information to anyone who inquiries about it. Indeed, blockchain technology offers two important values: traceability and transparency. One possible scenario is that there are two different online platforms such as eBay and Amazon, and users have two accounts (eBay account and Amazon account). The blockchain-trust-station can be informed by the required information to update the trust score of a specific user after performing a transaction on one platform. If a new transaction needs to be performed on any platform, people can learn about the reputation of users even if their previous transactions occurred on other platforms. In this case, participants and the operators of the platforms as well can track the users' behavior, and this is clear transparent since anyone can ask for trust information.

Cold-Start problem is another issue that the proposed work addresses. One of the principal shortcomings of the reputation-based trust model is a cold-start problem where a newcomer joins a community and has no prior interactions/no history to compute the reputation. The majority of the available reputation-based trust models do not take into account this problem. The main point is that the solution of the cold-start problem cannot be injected into the trust model which means it needs to be addressed separately. Therefore, our work builds a sub-model that aims to address the cold-start problem, but it is only applicable to e-commerce communities.

Another additional significance is that the model operator can regulate some of the model's parameters/factors, depending upon their preferences. For instance, one operator may decide to assign higher weights to the most recent transactions and to assign lower weights to the old transactions to increase or decrease the power of the transactions. Additionally, the threshold of the trust score may differ from one to another. The application may offer some flexibility to the user to set his/her own thresholds to categorize users as trusted or distrusted in the case of binary evaluation or to categorize users as trusted, semi-trusted, distrusted in the case of multiple evaluations. Uncharitable peers may enforce a higher threshold, while lenient peers may choose a lower threshold.

1.4. Research Questions and Hypotheses

This thesis will have three research questions designed to examine our developed model. These research questions and their hypotheses are given below.

Research Question 1: Does the TrustMe model mitigate the problem of Biased/Malicious Reviews?

Hypothesis 1: TrustMe model mitigates the problem of Biased/Malicious Reviews.

Research Question 2: Does the developed model alleviate the problem of cold start? (*Postponed for future work*)

Hypothesis 2: The developed model alleviates the problem of cold-start. (*Postponed for future work*)

Research Question 3: Does the blockchain-based trust model efficiently manage multiple trusts and alleviate the impact of malicious attacks?

Hypothesis 3: The blockchain-based trust model efficiently manages multiple trusts and alleviates the impact of malicious attacks.

1.5. Motivations of the Research

A large number of trust models have been proposed to compute the trust score of peers [6, 8, 7, 18]. These models aimed to analyze all previous transactions done between peers in order to distinguish between dishonest behavior and honest behavior of a peer and then compute its trust score. Miscalculation of the trust score is a central problem of computing reputations [11]. This can be true when measuring the trustworthiness of peers by simply finding the overall reputation score by aggregating a peer's ratings and thereafter finding the average. Unfortunately, there is no way for such a system to distinguish between a peer who consistently receives a certain rating and a peer whose behavior has resulted in a variety of negative and positive reviews. Likewise, the number of reviewers is important in such a system, since, for example, a peer with little good feedback in total can outshine a peer with thousands of mixed reviews.

Furthermore, running a verification of peers' identities is a key factor in building a reliable trust model. Unfortunately, it is easy for malicious users to create multiple accounts with fake identities and use them to increase their own reputation or harm other users. Equally as important, because of the increased number of collusion and Sybil attacks, it is extremely problematic to consider only feedback as the single factor to measure the trust of a peer. There is a need to improves the kinds of reputation systems by introducing a trust model that incorporates several factors to comprehensively measure the trustworthiness of a peer.

Fraud is increasing in peer to peer environment, especially in e-commerce. Reputation systems in e-commerce are at risk of malicious attacks, such as campaigns to falsely upgrade or degrade one user's reputation. In fact, in 2014, the number of fraud attempts was 1.39%, while in 2015, the number increase to 1.49%. In 2017, the global retail fraud attempts grew by nearly 31% according to ACI Worldwide [19]. Also, the fraudulent transactions are increasing during the holiday season such as Christmas, Thanksgiving (2%), Black Friday. It is stated in [20] that sales on Amazon and eBay are among these transactions that susceptible to fraud with 69%. They further stated that the fraudulent transactions on mobile sales reach 64%, while the retailer-owned e-commerce sites have 55%. This implies that fraud affects all types of e-commerce and can cause business failure if not handled correctly.

1.6. Research Methodology

This section briefly describes the research methodologies and what types of experiments were conducted to evaluate the work under study.

1.6.1. TrustMe Model Evaluation

We performed initial experiments to evaluate the promise of the proposed trust model "TrustMe." The method of this research will go through three phases. The first phase focuses on applying some analysis techniques ranged from unsupervised approaches, supervised approaches to statistical analysis. The second phase focuses on evaluating the TrustMe model in terms of bias and malicious attacks by conducting a simulation experiment and using randomly generated data based on some settings. In our study, we synthesize the available dataset by adding some extra information that is needed to represent some factors in the proposed model. The third phase includes designing an experiment that is based upon the assumption of using blockchain. This experiment will mimic the use of blockchain and study its effectiveness in reducing malicious attacks and managing online identities.

1.6.2. Phase-1: Experimental Analysis

- *Density-Based Clustering*: evaluating the goodness of the produced clusters.
- *Graph Analysis*: selecting some use cases and study the characteristics of these cases. Then, clarifying some connections between being trusted or distrusted and the characteristics of each user.
- *Correlation Analysis*: studying the correlation between each factor and the generated trust score to see how each factor influences the produced score as well as studying at what rate these factors are correlated to each other.

1.6.3. Phase-2: Experimental Evaluation

We are designing a simulation that mimics a set of experiments to evaluate the TrustMe model and show its effectiveness and benefits. The first set of experiments evaluates TrustMe in terms of its accuracy. The second set of experiments demonstrates the benefit of the TrustMe model when it is used in a distributed community, for instance.

1.6.4. Phase-3: Experimental Evaluation of Blockchain

This phase will include designing an experiment that is based upon the assumption of using blockchain. This experiment will mimic the use of blockchain and study its effectiveness in reducing malicious attacks and managing a cross-community trust.

1.6.4.1. Simulation Setup

Before starting the simulation process, we need to prepare and set some sub-models to be able to conduct the experiment. These models include a community model, trust model, evaluation model, threat model, and transaction model in conjunction with some parameters related to the simulation setup.

1.6.4.2. Community Model

The duty of this model is to generate a number of actors *NOA* to shape the community under study. Some available datasets are used since they represent real transactions. The actors of the given dataset are divided into three groups. The first step will start with the first group as an initial community plus injecting this community with a special group of untrustworthy users m_u. The next step appends the second group of the actors by connecting them to the first group depending upon the real information. The final step will add the rest, and in all cases, a group of malicious actors will be injected. *Mbr* indicates the rate of behaving maliciously as the untrustworthy actors may behave honestly for a while and then behave maliciously.

1.6.4.3. Threat Model

The major threat usually comes from a malicious actor by providing a dishonest evaluation to their peers. In order to camouflage their malicious behavior, untrustworthy users tend to perform a set of transactions in an honest way to build an attractive reputation waiting for a decisive moment to carry out their attacks. This model mainly mimics simple attacks where actors behave maliciously overtime after performing transactions. Nevertheless, a malicious campaign attack can be conducted as well by launching a set of simple attacks but with more parameters such as time frame. A list of possible threats is provided in chapter three. The volume of risk that threatens the community is captured by the overall malicious behavior percentage:

$$MB_{avg} = (Uar * NOA) * Mbr$$

1.6.4.4. Transaction Model

The transactions and ratings are already established in the given datasets. However, since the users are divided into groups, transactions need to be initiated correctly after appending each group. Untrustworthy users will provide ratings relying upon their plan. For instance, some untrustworthy users may want to harm the reputation of their target, while others want to improve the reputation of their target. The malicious behaviors as stated above occur based on the malicious attack rate Mbr.

1.6.4.5. Evaluation Model

The evaluation model simply solicitudes about specifying the assessment criteria to evaluate actors. Once a transaction completed, the participated actors may want to evaluate one another through a mechanism of evaluation. The rating model uses rating values range between [-10, +10] as -10 indicates an ultimately negative experience and +10 indicates an ultimately positive experience. If a trustworthy actor involves in a transaction with an untrustworthy actor ended up with dishonest behavior, the trustworthy actor can provide a negative rating against the untrustworthy actor.

1.6.4.6. Trust Model

The trust model implements three trust models (TrustMe, PeerTrust, & RawMean) for comparison purposes. This component produces trust scores based on each model’s design, and as a result, each actor will get three different trust scores. Classifying actors using crisp evaluation (trustworthy or untrustworthy) is according to three different trust models and three different thresholds.

2. TRUST AND TRUST MODEL

"Trust is a social good to be protected just as much as the air we breathe or the water we drink. When it is damaged, the community as a whole suffers; and when it is destroyed, societies falter and collapse," Bok, 1978 [21].

2.1. Trust and Reputation

The concept of trust is pervasive in human societies. By watching our daily lives, people can see how they trust the school bus to pick up their kids for the school, how they trust the doctor when he/she prescribes medication, or how they trust the police officer to protect their communities. There are a massive number of examples connected to trust.

Trust is defined as a relationship between two parties when one party (trustor) is willing to reckon upon the behavior of another party (trustee) [7]. As the trustor has no control over the action of the trustee, an uncertainty level will appear as a sign of potential risk involved in the interaction. In a social context, trust can be attributed between people, people and objects, and social groups. Gambetta [22] defines trust as a level of subjective probability with an expectation that an agent evaluates other agents before executing the desired action. This implies that trust is present before performing the interaction, and it is calculated due to the trustor's expectations.

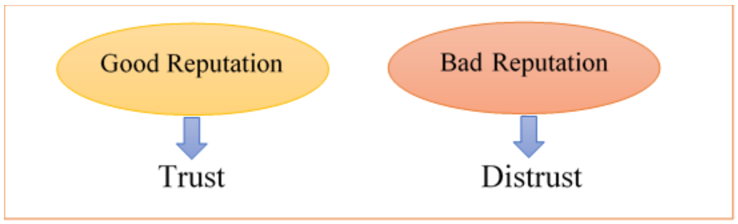


Figure 1. Trust and Reputation.

Reputation is another important term that differs from trust but highly relevant to it. Simply, when a person has a good reputation, others can trust him/her, while a person who has a bad reputation, others cannot trust him/her. Authors in [23] define reputation as "an expectation about an agent’s behavior based on information about or observations of its past behavior.” Therefore, reputation can be employed in establishing trust.

2.2. Trust and Agent

Traditional software is designed in advance to deliver some functionalities or to achieve a specific goal, and this software follows a number of predefined steps and instructions to perform these functionalities and to fulfill the assigned goal. In contrast to traditional software, agents are considered smart software applications that might require a minimum human involvement. Agents should be designed to find its way to achieve the assigned goal without telling them how exactly to achieve this goal [24]. Agents can interact with each other in multiagent systems (MAS). In Open MAS, agents are usually deemed distributed autonomous or semi-autonomous entities that might behave and make decisions by themselves. These agents can interact and cooperate together to complete an assigned task, but they are regularly unrelated and unknown to each other [25]. Autonomous or semiautonomous agents have the ability to interact, move, adapt, and evolve due to changes in their environments or due to some circumstances. However, this kind of system brings some risks since the malicious behavior could appear at any time from untrusted agents that their goals might be to fail or change the desired goal for the victim agent. Since agents have the ability of mobility, the threat level could highly be grown.

Moreover, in multiagent systems, a number of protocols and mechanisms organizes the interaction among these agents such as resource allocation, stopping agents from altering each other, etc., and these protocols might impose some rules to control the environment [26, 27]. Therefore, agents need to have a decision-maker component to decide with which they need to interact to execute their plan and receive the required service. Enough information about the corresponding agent, the environment, and other factors is required for an agent to be able to make a proper decision in order to trust the corresponding agent or not. However, some restrictions could lead to a situation when one agent cannot receive a suitable amount of information about the targeted agent, so the agent will be in a status of uncertainty to make a proper decision [27].

To solve the issue of uncertainty, agents need to trust each other. Trust models' objective is to guide agents in making appropriate decisions to establish trust relationships and then interact. Nevertheless, trust models presuppose agents to collect some information about the corresponding agents which could be gained through direct interactions or indirect interactions. With direct interactions, agents can learn, evolve, and collect some information about the corresponding agent that dealt with and then to be able to shape a trust relationship with it. The agent must be able to handle some risks such as dealing with dishonest or malicious agents for the first time. In indirect interactions, agents need to develop methods to gain information from other agents that are usually trusted because of previous transactions and interactions. A variety of issues could appear (i.e., lack of information and contradicted information).

Jansen & Karygiannis in [28] divide the security threats in mobile agents into three taxonomies: agent-to-platform, agent-to-agent, and platform-to-agent. The Agent-to-agent category, for instance, lists a collection of threats including masquerading, unauthorized access, denial of service attack, and repudiation. A masquerading agent might conceal its identity or use fake identity in order to deceive other agents that are involved with it in an interaction. In an e-commerce community, a masquerading agent may exploit the identity of another agent who has a good reputation in delivering services or products. For the sake of persuading a victim agent, the masquerading agent may use this identity to obtain the victim's private data and some important information such as credit card numbers. In this way, two entities will be harmed and negatively affected: the agent with a good reputation and the victim agent. Denial of service attack is another threat in which a malicious or untrusted agent might attack the targeted agent or host agent by requesting the same provided service voluminously at a certain time which causes preventing the other agents from receiving this service when they need it. An additional possible threat is a repudiation that could occur in this kind of system. This threat happens when two agents involved in the same transaction or communication, and then one agent controverts that this process occurred. This could cause some problems, but a host agent can prohibit agents from conducting repudiation without such as preserving the records of the process that has been done to help to resolve this issue. These records could be helpful in developing trust models to rate agents based on their histories and reputations. Unauthorized access is also a problem with which malicious agents can exploit some defects in the platform of a host agent. The host agent might have no efficient mechanism to secure the agents uses its platform. Thus, a malicious agent may be capable of intruding on another agent through accessing its data and/or code by using its public methods (e.g., attempt buffer overflow, reset to the initial state, etc.). Altering the agent code, for instance, will absolutely change the agent behavior, and as a result, a benign agent may become a malicious agent, and the damage could be disastrous.

Building a solid trust instrument requires to reveal and understand more issues and mechanisms that are used in agent systems. Such a mechanism is how agents delegate tasks to each other which is one mechanism that should be known to establish an appropriate trust relationship. The common types of the delegation are weak and strong delegations. An agent x could assign a weak delegation to an agent y without informing agent y. A simple example of weak delegation is stated in [29], a hunter is ready to shoot at a flying bird, and the hunter assumed that the bird would continue flying in the same path and direction. He is targeting the likelihood position that the bird will move to it in a few moments. By following his plan, the hunter is delegating action to the bird. Due to the lack of knowledge of this interaction, the bird will collaborate with the hunter's plan, and the bird will perform the required action. and this issue is primarily relevant to how the agent is secure. A similar scenario to the hunter and the bird could happen but maliciously, when an attacker agent is anticipating that some vulnerabilities could exist with a victim agent such as some weakness in the authentication process such as the targeted system or agent uses a very basic mechanism for passwords. In this type of delegation, to build trust relationships, a will-do belief is supposed to be present which is supported by willingness belief and persistence belief. Willingness belief means that the designed trust model takes into account that the delegated agent has the intention of performing the assigned task. In other words, the trust model designed to include modeling the other gent minds or behaviors. Persistence belief means that the delegated agent is steady in its behavior. If the delegated agent changes its behavior, it means that the assigned task could fail or deliver incorrect benefits. Since trust is essential to delegate agents, there are some situations in which trust is absent. This could occur when an agent has no choice or no information about the targeted agent, and the agent needs the offered service to finish its tasks (blind delegation), or the agent is not free to choose to delegate or not (coercive delegation) [29]. This kind of delegation is obviously a big challenge to handle when building trust models since agents may be delegated without their knowledge, However, in the strong delegation, the two agents that are involved are aware of the delegation. Since agent x wants to delegate agent y for performing one or more tasks, there might be some negotiation process including request, offer, and others to land up with an agreement between the two agents. Creating this agreement requires having a minimum level of trust to be established among agents because agents prefer to securely deal with trustworthy and reliable agents that are capable of delivering the requested service/benefits.

Autonomous agents should be granted with goals and beliefs to be able to execute and achieve their plan [29], and these agents could be cognitive. An author in [30] proposes a metaphor and refers to it as SDC metaphor: “view a human-computer pair (or any other pair) involved in a shared problem solved as a single cognitive system.” This metaphor indicates that human society and machine society establish an agent society whey they interact and participate in achieving a shared fulfillment of a task. A sophisticated cognitive agent might be equipped with machine learning models, including some data mining techniques, natural language processing, pattern recognition component, and other intelligent components that allow an agent to gain knowledge continuously from different sources by mining the acquired data and as a result, the agent evolves. The agent may look for some patterns, and after a while, the agent evolves and develops its behavior and responses based on the newly acquired knowledge. Hence, agents can learn and anticipate new problems and offering some solutions. For example, a cognitive agent could be designed for businesses such as banks to detect credit card frauds. By using machine learning models, the cognitive agent can recognize the regular and the preceding patterns of a customer’s behavior and outlays and then can identify uncommon activities that could occur. Feeding these agents, cognitive agents, with some factual scenarios and situations of embezzlement transactions and legal transactions, agents can shape and grow its knowledgebase and then develop its strategies to distinguish new patterns of fraudulent behaviors. However, without trust relationships, no one as human or agent can delegate tasks to autonomous agents because there is no enough information about how the cognitive agent will behave and evolve in long term agreements, or whether this agent is capable to protect itself from malicious attacks. For instance, what are the guarantees that after a while, an agent software does not behave maliciously and providing illogical and unfair sentences for court cases? In addition, in an e-commerce community, how can prevent an agent from being a thief after being evolved over time. In military operations, when an agent is responsible for guiding a missile to bomb its target, what are the guarantees that this agent will not change the assigned target due to changes in some environment variables. This change could occur by the agent itself or by malicious intruders.

Due to the aforementioned issues, trust clearly becomes a vital element that should be present in such an environment where the occurrence of risks and malicious attacks is highly possible. Setting up a trust relationship between a human and an agent or between two agents is very important to allow that trusted agent to act on behalf of the trustor in doing tasks. Because the trusted agent may be provided by sensitive and private data, trust models in such systems should offer some mechanisms to convince users that they are dealing with highly trustworthy agents, and their data is protected and secure.

2.3. Related Work

Feedback plays a significant role in measuring the trustworthiness of a user in most existing reputation systems. Using feedback alone, however, as a metric to measure the trustworthiness of users is not an effective approach [7]. The feedback parameter was recognized as a fair measurement with no prejudice in a series of trust models and reputation systems [31, 5]. There is a large list of reputation systems and mechanisms are presented for agent systems and online communities.

The most dominant reputation systems are those in Amazon [32] and eBay [33]. These models are created to be supervised and controlled by a central authority. They propose a seamless way to identify the reputation of users, and the user themselves can provide their own ratings against others. In eBay, the system uses five start system that counts the top two values (4 & 5) as +1 (positive feedback) and count the bottom two values (1 & 2) as -1 (negative feedback), and the middle value is counted as 0 (neutral). The models give a global single reputation value by aggregating the ratings over six or twelve months and calculating the average. These systems are trivial and highly vulnerable to malicious attacks. However, Amazon, in 2015, announced, with no much detail, that they replaced the old rating system by a new machine learning model that takes into account several factors.

An adaptive reputation-based trust model for P2P electronic communities called PeerTrust was proposed by Li Xiong [7]. Their model showed good results in terms of feasibility and effectiveness of measuring trust. The model utilizes a bunch of factors to calculate trust. These factors start from user satisfaction which represents the user rating value, the credibility of the user's feedback, the number of the transactions, the transaction context factor, and end with the community context factor. Even though the authors suggest a function of trust value or a personalized similarity as credibility measurements, the creditability factor is still a challenge that needs to be addressed clearly. The first suggested method is vulnerable to malicious users who can deceive the function and receive the high value of credibility by behaving honestly for a period of time and then behave maliciously. With the second choice, the similarity function may deliver an inaccurate value as some systems use the principle of a single rating between two users. Nevertheless, PeerTrust is deemed as one of the effective metrics in evaluating users' trustworthiness.

Sabater and Sierra introduced a decentralized trust model called Regret that utilizes the direct experience to represent subjective reputation [34]. This model is mainly designed for multi-agent systems and incorporates direct trust, witness reputation, and neighborhood reputation components. Regret exploits three information sources to be injected into the model: individual dimension representing direct experience between two peers, ontological dimension called

Sociogram, and social dimension representing third parties. Regret also applied the recency function that gives high weight to the recent ratings. The advantage of using a Sociogram allows the model to represent different social relationships suchlike competition and cooperation. Notwithstanding, this model does not mention the process of identifying and locating witnesses in the network.

Huynh et al. proposed a trust model called FIRE, which is inspired by the Regret model [8]. FIRE consists of an interaction trust that represents the direct interactions between two parties, role-based trust, witness reputation. Witness reputation represents the opinion of other agents (witnesses) about one agent's through monitoring its behavior and certified reputation components that represent previous ratings received from the agent's partner after performing some interactions in the past. These components coalesce together to produce trust scores for agents. The direct trust component of Regret is employed in FIRE, taking into account the temporal information by using a special recency function that differs from the function used in Regret. The duty of this function is to calculate the age of interaction. However, trust models that mount direct interaction in the formula are usually vulnerable to con-man attacks known as a confidence trick in which an attacker pretends to be honest in several interactions to fabricate a high trust score. Thereafter, when an opportunity of a high-risk interaction appears, the attack swindles the victim. Con-man attacks could occur in a cyclic form as the attacker maintains a good reputation [9]. Malicious agents can attack FIRE and Regret models through using a con-man attack because the two models do not take into consideration the malicious behavior of witnesses as well as a conspiracy among malicious agents. Salehi-Abari and White in [10] conducted simulation experiments against FIRE, Regret, and other reputation models to evaluate if they are con-man resistance or not. The study proves that when the malicious agent is cognizant of the settings of the model parameters/variables, the FIRE model can become susceptible to cyclic attacks. Additionally, even though the malicious agent has no comprehension about the parameters of the Regret model, it can be susceptible as well.

Carboni [35] proposed a decentralized and distributed feedback management system that can be built on top of the Bitcoin blockchain. Yet, the proposed system has no evidence to support collusion resistance. Collusion resistance is the ability to prevent malicious users from providing misleading feedback or using fake identities to build a good reputation for their stakeholders [18].

The mechanisms of these models extract the rating values based on computing the trust itself and neglect the possibility that the ratings can be either positive or negative, with the assumption that both categories have different properties and impact. This possibility is handled in the Trust-Me model. Moreover, only a single rating exists between two users, which means one user can receive one rating from each user, and this value can be updated without recording the previous rates. Trust-Me, in general, attempts to lessen the impact of malicious users and cyberattacks that target the reputation-based trust model. This work is a step toward building a comprehensive, reliable trust model that consists of multiple sub-models that can handle several malicious attacks and deliver precise trust scores.

2.4. The New Model “TrustMe”

By virtue of the claimed motivations, the TrustMe model is a reputation-based trust model developed to evaluate the trustworthiness of an individual/entity. This model is equipped with a list of vital factors that exploit the users’ ratings that are received after carrying out an interaction as a reputation measure. However, we believe that there is no single trust model that can handle all possible situations and solve all related problems such as the cold-start problem. Figure 1 shows the general architecture of a trust center that consists of multiple sub-models. Every sub-model is dedicated to addressing a set of issues that are relevant to trust and reputation.

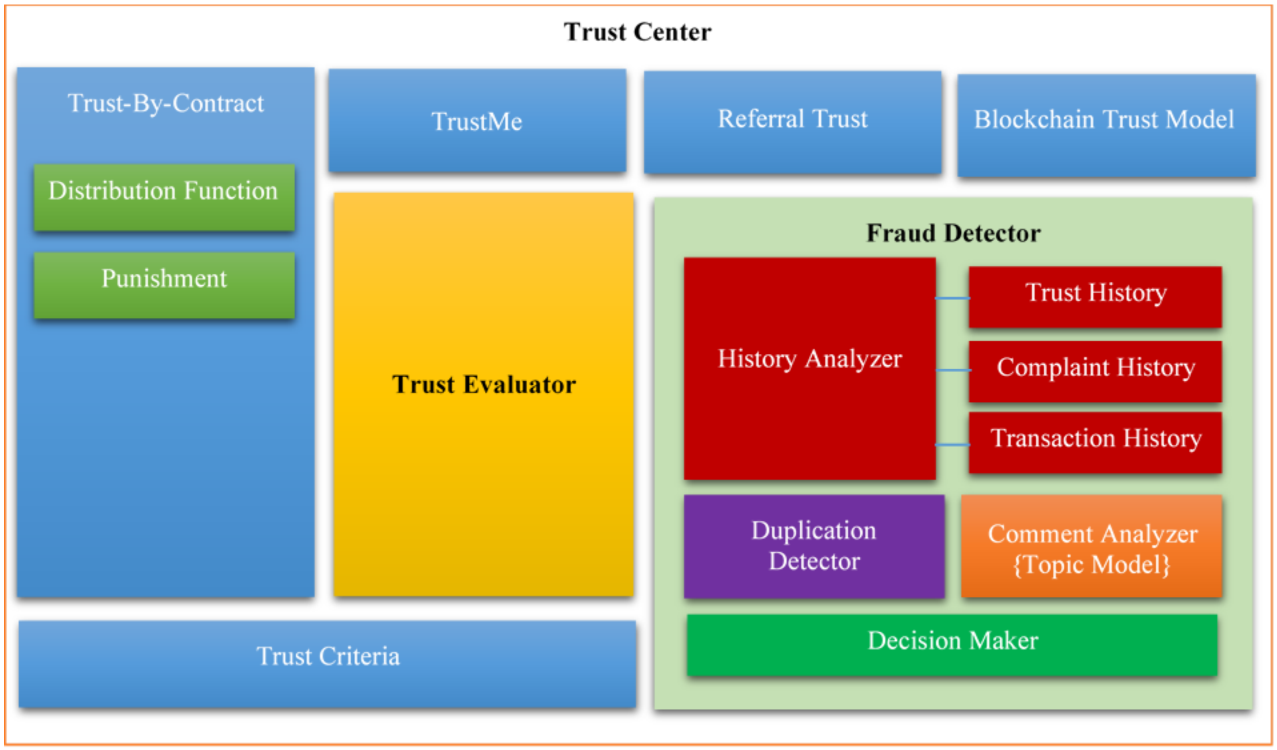


Figure 2. Trust Center Architecture.

In this section, trust factors are introduced and followed by a discussion of the trust metric describing how these factors integrate to address some trust issues. A trust modeling and computational framework that can be distributed, implemented, and kept current at scale is a massive task. Most trust models target P2P networks that can be seen as social networks where edges map out the relationships between peers. The edges may hold weights that are commonly employed to represent ratings. By means of performing interactions/transactions, those edges become dynamic and evolve over time. They are also relative and respective, as peers may trust or distrust one another at fluctuating and networked levels. The outcome of this model is typically a global trust score that is linked to a peer/user. This score mirrors the experiences of all peers that interacted with the holder of the trust score. The described trust model is called “TrustMe”; this model is reputation-based and consists of several factors. TrustMe model can be adapted to accommodate many varying contexts in cyberspace. In systematic and formal terms, this model copes with peers/entities and interactions denoted as follows: $e \in E \equiv \{e_1, e_2, \dots, e_n\}$, and $i \in I \equiv \{i_1, i_2, \dots, i_m\}$. Entity *e* can be viewed as a composition of subsets of popularity, and neighbor, whereas interaction *i* is a composition of rating, timestamp, and context subsets.

$$e \equiv \{P \cup D\} \equiv \{\{p_1, p_2, \dots, p_n\} \cup \{d_1, d_2, \dots, d_n\}\}$$

$$i \equiv \{R \cup H \cup C\} \equiv \{\{r_1, r_2, \dots, r_m\} \cup \{h_1, h_2, \dots, h_m\} \cup \{c_1, c_2, \dots, c_m\}\}$$

$$r \equiv \{r_{e_i}\} \qquad \text{for one-way rating}$$

$$r \equiv \{r_{e_i}, r_{e_j}\} \qquad \text{for two-way rating}$$

These factors can be measured with a value that falls between a minimum and maximum boundary. For example, the minimum value may represent the ultimate dissatisfaction, and the maximum value represents the ultimate satisfaction. The Popularity set represents the level of the sociality of an actor in a given community. Its values range between $[0, 1] \equiv [completely\ unpopular, completely\ popular]$: $0 \leq p_i \leq 1\ where\ i \in \{1, \dots, n\}$. So, trust (T) can be computed using the driving forces: e and i.

$$T \equiv \{t_1, t_2, \dots, t_n\}$$

$$T_e(E \cup I)$$

The trust attributes/factors collaborate together to build the metric of trust model and eventually generate a trust score.

2.4.1. Trust Factors (Independent Variables)

Trust factors are described as independent variables that shape the trust model. The developed model consists of eight factors. These factors aimed to enhance the mechanism of evaluating trustworthiness:

- Number of edges for each node {# of ratings} (*d*)
- Positive received ratings (*PRR*)
- Positive given ratings (*PGR*)
- Negative received ratings (*NRR*)
- Negative given ratings (*NGR*)
- Transaction Volume (*tv*)
- History influence (*h*)
- Popularity factor (*p*)

In the rest of this section, we describe and illustrate the importance of these parameters. We formalize the factors and show how they incorporate in evaluating the trustworthiness of a peer/an individual.

2.4.1.1. Factor 1: Number of Edges for Each Node {# of Ratings}

Some users may have a higher number of ratings than others, so the user's trust score cannot be captured in a fair way when only applying a simple aggregation of the total number of ratings. Also, with the lack of identification, users can increase their rating by creating multiple accounts, performing transactions using their own accounts, and then rating transactions. The total number of received ratings for each user can be calculated by counting the degree of the node which is the total number of its neighbors. The formula of this factor is given as follows:

$$d_i = \sum_{j=0}^{n} A(i,j)$$

where *A* is the adjacency matrix; d_i denotes the degree of node *i* (number of ratings); *n* denotes the number of nodes in the network.

2.4.1.2. Factor 2: Positive Received Ratings (PRR)

PRR is a metric that measures the centralization of the positive received ratings used to minimize the influence of malicious behaviors. For instance, the bitcoin-OTC and bitcoin- Alpha rating systems allow only one rating per user-pair. This means that if user *A* gives a rating to user *B* for a transaction and wanted to rate him/her for a different transaction, the rating system would remove the previous rating and update it by the new rating. With this enforcement of rating, users are affected by malicious campaigns to attack users by replacing positive reviews with negative ones or vice versa. To account for this, the median positive received rating is an important factor that must be taken into consideration. By taking the median of positive ratings, it assists in avoiding the impact of outlier values, such as bots or malicious users. The formula for this factor is given as follows:

$$PRR(u_i) = median\,\{_{j=0}^{n}\, in.r^{+}(u_i, v_j)\}$$

2.4.1.3. Factor 3: Positive Given Ratings (PGR)

The median of positive given ratings is a metric that measures the centralization of the positive given ratings and is currently not used in this model but listed to be used in future work such as analyzing the user's behavior. This factor is highly useful for this purpose, and it also reduces the impact of outlier values that may exploit by malicious users. The formula for calculating the *PGR* factor is given as follow:

$$PGR(u_i) = median\,\{_{j=0}^{n}\, out.r^{+}(u_i, v_j)\}$$

2.4.1.4. Factor 4: Negative Received Ratings (NRR)

The median of negative ratings is a metric that scales the centralization of the negative received ratings. This factor is employed in the model and represents the negative feedback received for an interaction usually ascribable to inappropriate behavior. The formula for calculating the *NRR* factor is given as follows:

$$NRR(u_i) = median\,\{_{j=0}^{n}\, in.r^{-}\left(u_i, v_j\right)\}$$

2.4.1.5. Factor 5: Negative Given Ratings (NGR)

The median of negative given ratings is a measure for finding out the central value of the negative given ratings and depicts the level of bad behavior received in an interaction. *NGR* is neglected in this model to be employed in future work since it complements *PGR* in observing the user's behavior. The formula for calculating the *NGR* factor is given as follows:

$$NGR(u_i) = median\,\{_{j=0}^{n}\, out.r^{-}\left(u_i, v_j\right)\}$$

2.4.1.6. Factor 6: Transaction Volume (tv)

Transaction volume *tv* represents the size of a transaction, usually in the form of currency. In the context of business savvy, the transaction volume is an important element that cannot be ignored and should be involved in computing trust. This factor will be employed as an extra defense line that collaborates with other factors in minimizing the impact of malicious attacks. *tv* is a weight that maximizes or minimizes the influence of the provided feedback/rating. In case of having a transaction with a small amount, the power of the provided feedback will be small as well, and in case of having a transaction with a large amount, the power of the provided feedback will be large. If this factor is not required, it can be ignored by assigning one to *tv* so that all feedback will be treated equally.

2.4.1.7. Factor 7: The History Influence

The history factor is portrayed by decency function which computes the age of the rating. In many cases, to assess the user based on the persistent pattern of his or her behavior, it is logical to give very old rates a lower weight, while the most recent rates should receive a higher weight.

If the history factor is not necessary, it can be ignored by assigning the value "1.0" to a weighting factor *h*. For instance, if ratings are distributed in a period of one year, then weights can be sliced as follows {the range is between [0, 1] and the total = 1}:

```
    if (t ≥ 0m and t ≤ 3m) then H = 0.45
    if (t > 3m and t ≤ 6m) then H = 0.30
    if (t > 6m and t ≤ 9m) then H = 0.20
    if (t > 9m and t ≤ 12m) then H = 0.05
```

where *t* indicates timespan and *m* indicates months. This factor can be used as follows:

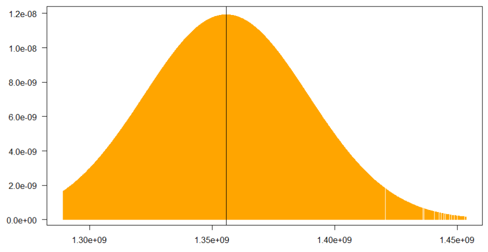
$$PRR(u_i) = \ median\,\{_{j=0}^{n}\, h\ * in.r^{+}\big(u_i, v_j\big)\}$$

Dividing the history of all ratings into several periods is challenging. Moreover, generating multiple thresholds is required and can be obtained by the user when he or she wants to identify their own preferences. Some users may prefer to increase the power or the weight of the most recent rates and reduce or ignore the oldest ones, usually based on some criteria. The slicing process can be done with the help of a Gaussian distribution function, with a threshold assigned to each slice. The following is the Gaussian distribution formula:

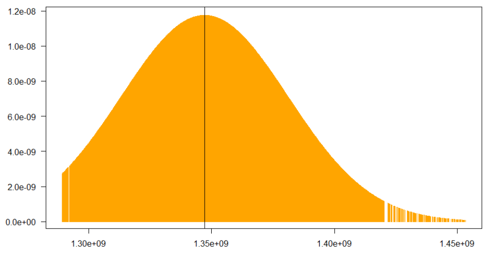
$$f_g(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-a)^2}{2\sigma^2}}$$

For example, the bitcoin-Alpha and bitcoin-OTC datasets are used in our experiments and have the following distribution after standardizing the time and then applying the Gaussian distribution. Bitcoin-Alpha is with mean = (1347115565.43) and standard deviation = (33949905.54), and Bitcoin-OTC is with mean = (1355711027.46) and standard deviation = (33447387.80). By finding the mean (μ) and the standard deviation (σ) of a single user, it is easy to generate several periods. Figure 3.a and 3.b show the time distribution of the two given datasets.

The number of slices can be increased or decreased based upon the preferences or based upon the rules of the model's operator. Also, the threshold values are logically dependent upon the users' preferences or agreed-upon rules. History weights were divided into five categories, and this is used in evaluating our model. These categories are listed in Table 1.



a: bitcoin-OTC dataset



b: bitcoin-Alpha dataset.

Figure 3. Time Distribution of the Two Datasets.

Table 1. Distribution Rules.

Rule	Weight
if $(t > \mu + 2\sigma)$	w_1
if $(t > \mu + \sigma$ and $t <= \mu + 2\sigma)$	w_2
if $(t > \mu - \sigma$ and $t <= \mu + \sigma)$	w_3
if $(t > \mu - 2\sigma$ and $t <= \mu - \sigma)$	w_4
if $(t <= \mu - 2\sigma)$	w_5

2.4.1.8. Factor 8: Popularity

Popularity is a metric that refers to how social a user is within a community. Popularity is evaluated by counting the number and quality of links to a user to measure how important he/she is in the community. The assumption is that most important user, who receives more links from other users, will be more influential than less active users, and the weight of their ratings will be raised in proportion to their popularity. The well-known Page-Rank algorithm is employed to compute the popularity factor. The formula is given as follows:

$$Pr(u_i) = \frac{1-d}{N} + d * \sum_{u_j \in M(u_i)} \frac{PR(u_j)}{L(u_j)}$$

Table 2 summarizes the variables that are incorporated to shape the dimensions of the proposed model. The values of these variables are regularly obtained by performing interactions/transactions between parties/users. Therefore, the generated trust scores are dynamic due to more interactions performed with different values assigned to the employed factors, and this may raise or lower the trust depending upon the interaction satisfaction.

Table 2. Summary of the Factors of the Proposed Model.

Factor	Description	Range
h	indicates the prevalence of the rating historically (the age of the rating/timestamp).	[0, 1]
p	indicates the level of sociality (how the user is popular in a community).	[0, 1]
$\boldsymbol{d^+}$	denotes the total number of the positive received ratings.	[0, 1]
$\boldsymbol{d^-}$	denotes the total number of the negative received ratings.	[0, 1]
tv	denotes the transaction volume (context); to ignore its influence, let *tv=1*.	[$Min, $Max]
d	denotes the total number of ratings that one user receives. the total number of incoming ratings for a user representing the number of positive ratings and the number of negative ratings.	[0, Max]
$\boldsymbol{\theta}$	a weight for the second part of the equation; determining the influence level of this part.	[0, 1]
$\boldsymbol{\gamma}$	a weight for the second part of the equation; determining the influence level of this part.	[0, 1]

2.4.2. Trust Score (Dependent Variable)

The outcome of the proposed model is a trust score, calculated by the provided factors that form the trust metric. To measure the trust score of an individual, the model needs to extract the required data for each factor form the working environment. The factor *PRR* depends on the positive given rating values; *NRR* depends on the negative given rating values, and so forth. A high number of interactions improves the accuracy of the produced trust score. The variations in the trust factors might increase or decrease the trust score.

2.4.3. TrustMe Model

In this section, the overall model is presented. This model is inspired by the PeerTrust model with some differences. A user's trustworthiness is established by evaluating the user employing all factors presented above. The formula of the model is given as follows:

$$T(e_i) = \left(\theta * d^+ * \left(median\, \{_{j=0}^{d^+} tv_j * p_j * h_j * r^+(e_i, e_j)\}\right) + (\gamma * d^- * (median\, \{_{j=0}^{d^-} tv_j * p_j * h_j * r^-(e_i, e_j)\}))\right)$$

where tv_j indicates the weight of the transaction volume of a transaction between *i and j*; p_j denotes popularity value for user *j*; d^+: the number of users who gave positive ratings to e_j; d^-: the number of the users who gave negative ratings to e_j. In order to use the simple form of the model, the history factor can be ignored by assigning 1.0 to *h* and assigning 1.0 to the popularity factor *p*

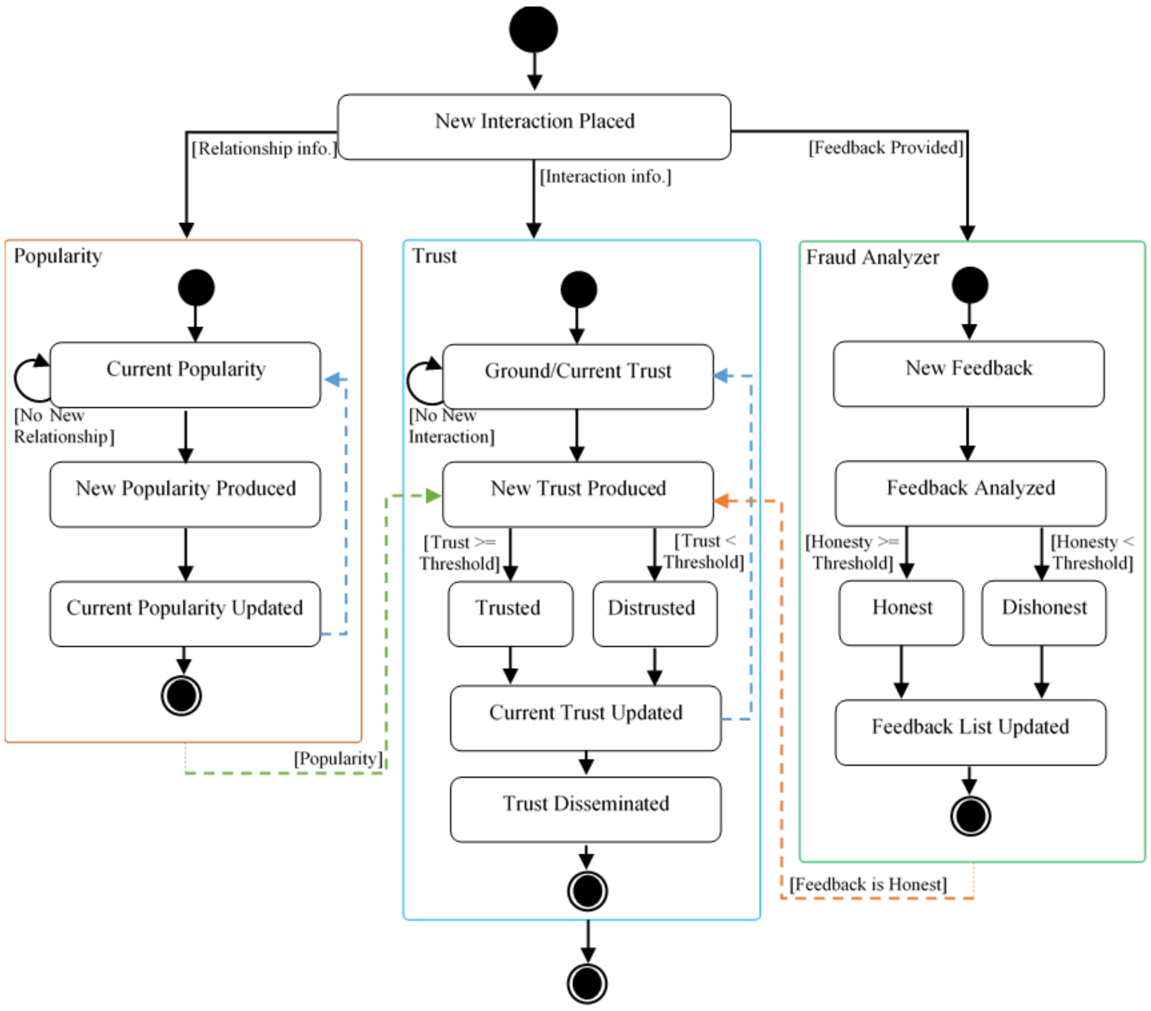


Figure 4. A State Transition Diagram of the TrustMe Model.

In order to guarantee the efficiency of a trust model, a fraud/deception model needs to be plugged into the equation. The fraud model is a complementary model that aims to filter out dishonest feedback based on predefined criteria. Figure 4 shows a general state transition diagram of the trust model. The figure depicts how the fraud component interacts with the trust component to filter out dishonest ratings/feedback. The diagram also explains the dynamism of trust and popularity when an interaction takes place. A new trust score will be generated and updated after the ratings are filtered out by the fraud analyzer, whereas the popularity value will be updated if a new relationship is established. Additionally, the decision of categorizing a peer as trusted or distrusted is based upon the value of a threshold so that $if\ (T \geq Trust\ threshold)\ then\ state = trusted$. The diagram is divided into three sections (popularity, trust, and fraud analyzer). Each section portrays the change of states from one to another according to specified rules.

The TrustMe model provides an indication of the computational modeling and work involved in invoking and maintaining a near real-time trust model that can be fully decentralized. Additionally, the TrustMe model separates the positive evaluations and the negative evaluations to help to reduce the impact of dishonestly upgrading or degrading the trust score. When some actors attempt to increase the trust score, they will provide positive ratings to lower the influence of negative ratings, but with such separation, their plan hardly succeed. Add to that, other factors such as popularity and history that make the goal of such dishonest behavior challenging to accomplish. The weights θ & γ can be employed as rewards and punishments.

3. BIAS ISSUE IN TRUST-BASED SYSTEMS: MALICIOUS AND SPONTANEOUS ATTACKS

3.1. Introduction

Online communities involve an abundant number of people who interact with each other. e-commerce web sites, for instance, is one kind of these communities where users interact as sellers and buyers. Thus, trust has manifested on the horizon as a crucial element in these communities. Trust refers to a relationship when One party (trustor) is willing to reckon upon the behavior of another party (trustee) [7]. In another manner, the trustor's dependence is upon the trustee's behavior. Trust has a big impact on the success of a business. Trust is a factor determining the success of a business through evoking the customer's trust.

Feedback provided by reviewers is usually used to build the reputation and trust of an individual or an entity. A reputation system has become critical to reinforce trust and lessen risks. These systems monitor the history of users through gathering and aggregating review values and finally propagate the reputation [36]. Therefore, many researchers introduce trust models that are based on reputation (reputation-based trust models). The major intention of this kind of model is to assist in estimating the trustworthiness of an entity/individual employing users' feedback. Accordingly, the quality of future activities of a trustee can be expected [7, 37]. Another evident intention for trust models is mapping users to one of their cues/traits to be classified as honest or dishonest. Such models help to guide people making a proper decision when they decide to sell or buy goods using online platforms such as eBay and Amazon. This practice implicitly drives users to behave and tie with veracity, honesty, and probity which eventually establish a trusted environment, and also it can impede malicious behaviors [36].

Frivolous small biases in users’ feedback could negatively affect the computation process of trust values. If the trust model does not have a mechanism addressing such an issue, this bias may substantially weaken the accuracy of the trust model which makes its presence and absence more alike. Users may misconduct when they provide feedback on others and generate false information, and this behavior may distort the trust level of an individual or an entity. The pervasiveness of bogus reviewers who gives misleading feedback is apparent, but it is difficult to disclose because some reviews are provided in a professional, malevolent way that shows them factual and logical. These practices create what is called bias. Behind this bias can perhaps be a malicious attack or perhaps spontaneous misbehavior. Designing a trust model, hence, that can abolish or macerate the impact of bias, is a major challenge. For the sake of simplicity, studying and dealing with the process of detecting biased reviews should be conducted separately because it requires a canny mechanism and considerable ingenuity. Ultimately, this mechanism can be injected into the trust model to handle bias problem.

3.2. Bias

Cambridge dictionary defines bias as “the action of supporting or opposing a particular person or thing in an unfair way, because of allowing personal opinions to influence your judgment (2018).” Whilst Sackett defined bias as “any process at any stage of inference which tends to produce results or conclusions that differ systematically from the truth” [38]. The trust model becomes a paramount player that helps in sieving the inundate of information and supplying people with metrics that facilitate and ease the process of decision-making. The information source such as online reviews that are used in reputation-based trust models might be biased/attacked and consequently twist a user’s opinion into a specific direction. It is essential to avoid dealing with detrimental reviews or information in order to hinder misleading users. Remedying and determining the causes and the occurrences of bias, therefore, is indispensable to collect unbiased information and calculate trust correctly and more precisely. Generally, bias can be divided into two main categories: spontaneous bias and malicious bias. Both categories can delude users through feeding the trust model by contaminated reviews.

3.3. Dual-Process Theory

Basically, bias can be divided into two major categories: spontaneous bias and malicious bias/attacks. Understanding the behavior of users whether they provide explicit or implicit biased reviews magnetizes the attention toward Dual-Process Theory. In this theory, psychologists have figured out that humans have two different ways of thinking [39, 40, 41]. Dual-process theory sheds light on the process of how people behave as a reaction to the received information. The goal behind applying dual-process theory in the context of bias is to crystallize how reviewers became biased when giving their ratings. Since many psychologists have developed this theory across different cognitive domains, nearly all of them asserts that the characteristics of each way of thinking are very similar [39, 40]. On the first hand, the premier way of thinking is called system 1, heuristic system, implicit system, etc. Some of the characteristics of system 1 are unconscious reasoning (intuition, subconscious), mostly linked to emotions, implicit, automatic, low effort, and fast. On the other hand, the other way of thinking is called system 2, systematic, explicit system, etc. The characteristics of this way are conscious reasoning, mostly disconnected from emotions, explicit, controlled, high effort, and slow [42, 39, 40, 41].

The following example provides a simple illustration of the difference between the two ways of thinking: the first system corresponds to when people think of calculating a very simple addition or subtraction such as *1+1 = 2 or 1-1 = 0*. Whereas the second system corresponds to when people think of calculating a more complicated equation that stimulates conscious thinking such as $\sqrt[3]{7889}/_{976} + 645$. The first equation requires low effort and will automatically be solved, while the second equation requires high effort and solving it will be slower than solving the first equation. The following section briefly discusses two renowned dual-process theories: Elaboration likelihood model, and the heuristic-systematic model will be discussed briefly as they are renowned dual-process theories.

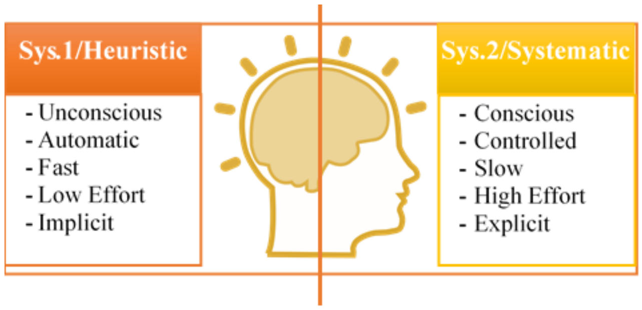


Figure 5. Dual-Process Theory.

3.3.1. Elaboration Likelihood Model (ELM)

ELM is a dual-process theory of information processing developed by Richard Petty and John Cacioppo in 1980 [43]. ELM introduces two prominent paths for information processing which are the central route and the peripheral route aiming to portray the attitude changes [44, 43]. Choosing either route reckons on the individual’s elaboration likelihood. If the elaboration likelihood is high, the central route will be utilized, and if the elaboration likelihood is low, the peripheral route will be utilized [44, 43, 45]. Further, authors in [43]state that the capability of an individual to appraise information, his personal relevance, and his motivation affect the elaboration likelihood. For instance, people who have a desire, ability, and intrinsic motivation to elaborate are supposedly excess their use of cognitive resources in order to assess the information quality, and this connects to the central route.

3.3.2. Heuristic-Systematic Model (HSM)

HSM is a communication model developed by Shelly Chaiken in 1980, and it argues that humans can handle messages in two different ways via employing heuristic mode or systematic mode [42]. Heuristic processing relies on retrieving judgmental rules stored in memory, and this process leads to making a speedy decision disregarding some portion of the information. On the other hand, systematic processing relies on analyzing the message content and identifies its veracity by looking at relevant information and making a decision based on the analysis results [42, 46].

3.4. Types of Bias

Simply, bias can be divided into unintentional bias and intentional bias where unintentional bias is considered spontaneous and intentional bias is considered malicious. The two categories of bias can be named as normal and abnormal bias, spontaneous and malicious bias, spontaneous and malicious bias, implicit and explicit bias, etc. Offering a precise trust preferably requires determining whether the detected bias is intentional or unintentional. Reasonably, reviewers who have been caught giving intentionally biased ratings should be penalized by reducing their trust value severely than reviewers who gave unintentionally biased ratings.

3.4.1. Spontaneous Bias/Attack

It is a bias with no intention to cause sabotage or harm. Spontaneous bias can be called as an innocent bias and categorized as implicit bias. Since humans are impressionable, reviewers confer their ratings unconsciously without deliberate control, and this can be defined as common thinking errors that affect people's decision-making. The spontaneous attack clearly belongs to system 1 or heuristic processing in HSM model and belongs to peripheral route in ELM model because in such scenario, a reviewer tends to depend on general impression, moods, loyalties, etc., with no much effort, so his/her likelihood elaboration level will below. The following list shows some types of spontaneous attacks that are placed in users' ratings and affect the trust computation process.

3.4.1.1. Systematic Bias

It is the penchant of supporting a specific opinion repetitively and contrary to the real experience. Some research calls systematic bias a simple bias as in [47]. For example, systematic bias occurs if a buyer submits low or high ratings in a continuous manner and does not represent the genuine experience [47]. Such ratings possess traits that represent the reviewer's behavior, such as being unconscious, fast, low effort, and implicit, and this links to heuristic approach processing in HSM and likewise connects to the peripheral route in ELM since the elaboration likelihood is low [42, 39, 40].

3.4.1.2. Confirmation Bias

It is the tendency of confirming preceding opinions or beliefs through construing and reminiscing information that supports them and ignores the counteractive information. It is also called myside bias or confirmatory bias. Reviewers translate the information in a way that supports their positions, and this can be linked to overconfidence issue. For instance, when a buyer has chosen and bought a specific product based on some beliefs or his/her own interpretations, the buyer tends to supply a high rating in order to justify his/her purchase and to support his/her opinion. Further, they ignore any opposite information that weakens his/her opinion fleeing from confessing that he/she made an improper decision. In other cases, even though before the occurrence of the purchase, the majority of available ratings stands against the buyer's opinion, he/she ignores the opposite ratings and only looks at ratings that confirms his/her opinion.

3.4.1.3. Groupthink and Bandwagon Bias

It is the tendency of choosing the same direction as others or following others' actions regardless of the available proof. Authors in [47, 48, 49, 50] mention this phenomenon as sequential bias. This bias could happen when a reviewer gets influenced by others and acts according to their ratings by giving the same rating (get on the bandwagon) deviating from the proper decision.

3.4.1.4. Endowment Bias

It is also known as divestiture aversion or loss aversion. It is the penchant of averting losing a certain value in order to attain a similar value. Some people think that finding a way to gain ten dollars is preferable than losing them. In the rating advisory paradigm, buyers and sellers may mutually evaluate each other such as eBay, and everyone can provide his rating against the other. A buyer may give a high rating because he/she likewise wants the seller to give him/her high rating, and vice versa, which will decisively produce bias in the community. Reviewers likely become more loss averse when they experience obtaining more and more low ratings.

3.4.1.5. Focalism Bias

It is also known as the anchoring effect or the anchoring bias. It is the tendency of placing reliance intensively on an inceptive piece of information as a guide. The inceptive piece of information plays the role of anchor. Consequently, the process of decision-making is driven by this anchor, and it may skew the ensuing judgment. For example, due to the inability to handle things properly and handily, careless, clumsy users may excessively focus on some factors such as the period of the product warranty, which affects the subsequent rating while paying less attention to the other factors. Therefore, the anchor of this situation is the product warranty. The price of the product could be an anchor of a thrifty user. This bias is relatively different from one reviewer to another, and everyone establishes the rating on his own anchor.

3.4.1.6. Selective Reporting Bias

It is sometimes called under-reporting bias or reporting bias. It is the tendency of reporting part of the available information and hiding the big picture. Thus, the presented information does not reflect the actual reality due to the absence of a large fraction of the information. Such behavior increases the rate of uncertainty, and the buyer gets confused and makes decisions that are unsuitable for the buyer's situation. In various instances, buyers tend to refrain from providing ratings, which leads to swaying the process of decision making.

3.4.1.7. Cold-Start Bias

This type of bias occurs when a new user joins the community, and the system of the community has to establish an initial new trust value to the newcomer. Some systems provide a newcomer a zero-trust score, which some researchers consider this as unfair bias since people generally have some trust even without dealing with others. Other systems place the trust value of the new user in the middle (50%). If the system trust values fall between 0 and 1, the newcomer will get 0.5 trust value. Critics state that even this type is unfair bias because malicious users may open new accounts and easily obtain new trust value at this level, and their old ones might be below this value. Thus, this type of bias is largely controversial.

3.4.2. Malicious Attack/Bias

It is a bias with obvious premeditation to cause sabotage or harm to the victim. Malicious bias is categorized as an explicit bias. Malicious reviewers deliberately and consciously fabricate their given ratings to destroy the reputation of an individual or an entity, and the ratings are injected in a way to appear as authentic. This type of bias is difficult to detect since it is designed and preplanned with sober-minded thinking in order to appear genuine. Therefore, malicious bias belongs to system 2 or systematic processing in the HSM model and belongs to the central route in the ELM model. Here, the reviewers' likelihood elaboration is high because they usually study the crime scene and design an attack based on some readings that demonstrated the victim's situation. Malicious bias apparently entails motivation, desire, and ability to fulfill the goal of misleading other users. Those malicious reviewers are usually characterized by being conscious reasoning, mostly isolated from emotions, explicit, controlled, high effort, and slow in their thinking. Accordingly, malicious bias is indeed a malicious attack and threatens the trust computation. The following list shows some types of malicious biases that can be injected as feedback into the review system and skew the trust computation process.

3.4.2.1. Singular Malicious Attack/Bias

It can be called a basic malicious attack/bias. This type of attack is very trivial and simple. Its impact is not severely harmful if it is done in a solitary style. Notwithstanding, it can be harmful if it joins other types of attacks such as spontaneous or malicious attacks. It happens when a single dishonest buyer tries to inject a falsely rating against an honest seller by providing a negative rating. Another possible attack is when a dishonest buyer wants to support a seller for whatever reason by falsely providing a positive rating.

3.4.2.2. Sybil Attack

It is an attack that targets reputation systems and any system that is based on reputations with the intention of ruining them [51]. The name is inspired by a book titled Sybil, which discusses a case study of a woman named Sybil Dorsett with multiple personality disorders. Brian Zill introduced the name of this attack (Sybil attack) in 2002 at Microsoft research [52]. However, before 2002, Lawrence Detweiler introduced a threat called pseudospoofing in 1993, which is similar to Sybil attack [53], but it received no considerable attention. Sybil attack depends upon the basis of creating multiple pseudonymous identities (Sybils) aiming to tamper a reputation score [54, 52]. Consequently, a bunch of fake identities is controlled by a single entity. The power of the single entity is enlarged by the number of additional fake identities, so the extra the fake identities, the extra the impact will be. This power clearly affects the reputation of an individual/entity throughout performing multiple transactions and providing multiple ratings. Sybil attack may affect the victim identity and the functioning of the system as well. Undoubtedly, the Sybil attack increases the rate of bias in the given ratings and influencing the trust score as a consequence.

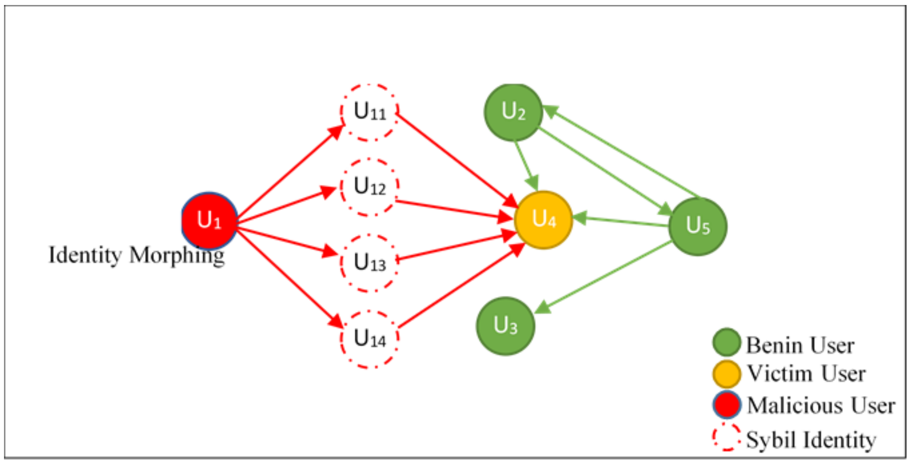


Figure 6. Sybil Attack Network.

3.4.2.3. Self-Promoting Attack

It is an attack wherein the trust of an attacker can deceptively be maximized by itself. The ability to achieve a successful self-promoting attack typically demands to comprehend the formulation of the operated model in order to affect the process of its algorithm [55]. Since some systems do not require interaction proof to allow reviewers to provide their ratings, in this case, the attacker gets a fertile environment to carry out his/her attack. Amazon and other e-commerce platforms have started providing a sign "*verified purchase*" to denote that the given evaluation is based on an actual transaction. However, no clear indication shows that the calculation of the reputation ignores unverified evaluations. A self-promoting attack can be carried out in a singular fashion (individual collusion) or a group fashion (collaboration collusion). Attackers can shape groups to improve their trustworthiness by conferring positive ratings to each other. The conspirators may perform some transactions with small amounts to make their given ratings verified. Sybil attack, additionally, can partake in a self-promoting attack in which the attacker creates multiple fake identities, then interact with and evaluates himself positively using these identities [52, 55].

3.4.2.4. Masquerade Attack

It is also called the impersonation attack. It is an attack where one entity successfully conceals behind the identity of another entity that ordinarily has a good reputation [28]. A masquerading entity can steal the credentials of a legitimate account or use a keylogger. The attacker may take advantage of a victim's over-trust in which trust surpasses the system capabilities, so the attacker can easily receive the victim's credentials, such as leaving the victim's device and account open for a while. The victims of impersonation attacks are probably users with high trust or a good reputation. This attack aims to get benefits from the good reputation of the victim for whatever purposes.

3.4.2.5. Malicious Programmed Behavior Attack (MPB Attack)

It is a prior prepared programmed attack consists of a miscellaneous assortment of independent and/or dependent cross-cutting malicious scenarios/scripts that regularly become woven together at the attack execution. The term is inspired by programmed behavior in psychology, which means that some human characteristics are programmed in advance like feeling hungry leads to eating, so people can eat without learning how to eat [56], and psychologists sometimes mention to programmed behavior as a response. MPB can be introduced as a collection of malicious scenarios, and each scenario represents a sequence of activities that the attack should carry out under certain conditions. Therefore, it is considered a highly effective attack due to its ability to cleverly bypass existing defense mechanisms. Forming a more advanced and complex attack could lead pernicious forces to build a machine learning model to generate a collection of witty malicious scenarios. Commonly, data represents the environment encircling a victim or target is injected into the model. This data can be obtained from multiple sources to generate a large set of possible scenarios to destroy the reputation of a victim. Although the malicious model is programmed before producing harmful courses of action, the model can learn from the proposed data and develop new more complex scenarios to be severely harmful. Let's assume that a group of people decided to launch a malicious campaign attack to harm the reputation or reducing the trust level of one seller. They can decide to apply the concept programmed behavior attack, and everyone in the group will be treated as an independent/dependent path that is responsible for generating and executing one or more specific scenarios. The tactics of each path/attacker can be executed separately with no need to interact with other paths unless there is a direct dependency.

- *Discriminatory Behavior Attack:* it is an attack that is based on discriminating between peers when it comes to choosing with whom the bargain will be [57]. For instance, a single user chooses only to deal with people who have high trust scores, so the rating weights of those people will be high. The weights affect the calculation of trust and improve the attacker's reputation. This kind of attack is commonly targeting some vulnerabilities in the reputation systems, and this happens when the attacker acquaints the formulation of the trust model.
- *Victims of a Victim Attack (VoV Attack):* it is a special version of indirect attacks. The indirect target is called a major victim, and the direct targets (minor victims)

are other users who interact with the major victim (victims of victim attack). This attack occurs when malicious users start harming the reputation of minor victims in order to severely harm the reputation of the main victim. When the minor victims' reputations get harmed and reduced, the influence of their feedback will diminish due to its low weight, and eventually, the reputation of the major victim gets harmed. *VoV* attack, in some positions, requires knowing the formulation of the model to create a craftier attack.

3.5. Norman's Theory

Since any attacker needs a plan or a design to perform a malicious attack specifically the craftiest ones. This leads the attention toward Norman's theory of action in order to understand how humans perform their actions, either maliciously or unmaliciously. Norman's introduced his theory about an individual's behavior and dissect the human action into seven stages as a structure of action: one for goals, three for execution, and three for evaluation [58]. The theory divides the stages into two main formulated sections: the gulf of execution and the gulf of evaluation, as shown in figure 7. The following is the Seven Stages of Action [58].

- Identifying the Goal
- Forming the intention
- Specifying an action
- Executing the action
- Perceiving the state of the world
- Interpreting the state of the world
- Evaluating the outcome.

3.5.1. The Gulf of Execution

This gulf focuses on the process of performing an action. After identifying the goal, the first step is initializing the intention and then transformed into a sequence of actions. The following three elements of this gulf.

- Forming the intention to Act.
- Translating the intention into a Sequence of Actions (specification).
- Applying and executing actions in the real world.

3.5.2. The Gulf of Evaluation

This gulf focuses on the process of evaluation. After finishing the last step in the gulf of execution, the process moves to the gulf of evaluation, and the next step is perceiving, followed by an interpretation, and eventually, evaluation. The following three elements of this gulf.

- Perceiving the world after the execution of the actions.
- Interpreting the perception of the world.
- Evaluating the interpretation.

3.5.3. Case Study of Applying Norman's Theory

The scenario is for malicious attacks to harm the reputation of a user in one online community, so the attacker needs to build a plan for this attack. Based on Norman's theory, the behavior could be as a series of actions as follow (shown in figure 8):

1. Identifying the goal: the goal is harming the reputation of a seller (victim).
2. Forming the intention: creating a malicious attack consists of a group of attackers to achieve the goal.
3. Specifying the intention, generating a sequence of actions:

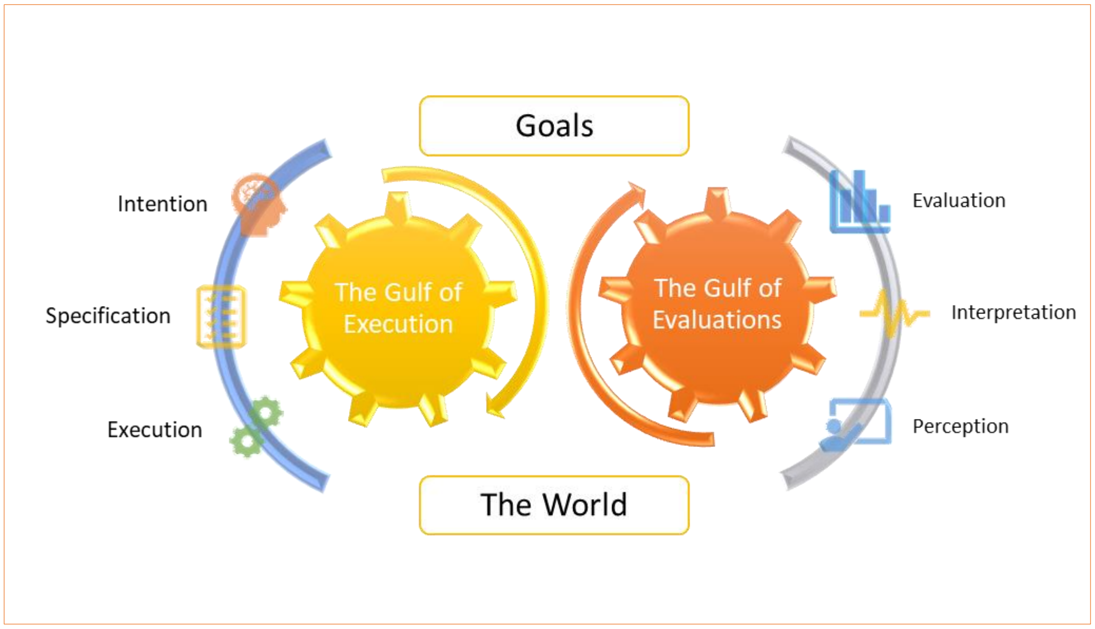


Figure 7. Normans' Theory of Action – Seven Stages.

a. Study and analyze the current status of the victim. For instance, what is the trust score? How many transactions does the victim perform?

b. Understand how the trust model is working: Let's take the eBay review system as an example since its metric is visible for all users.

c. eBay uses a raw mean metric, so the attackers can do some calculation to see how to attack the victim.

d. The attackers may decide to:

    i. launch a malicious campaign attack by buying cheap items and providing low ratings.

    ii. The attack should be carried out in seven days.

4. Executing the specified plan in the real world, which in our case is eBay and specifically the victim.

5. After bridging the gulf of execution, the process moves to the gulf of evaluation: perceiving the world and seeing what happened after executing the attack by reading the state of the victim after each attack, for example.
6. Interpreting the perception such as drawing some charts that show how the trust scores of the victim got affected over time.
7. Evaluating the interpretation by comparing the outcome to the intention and the identified goal.

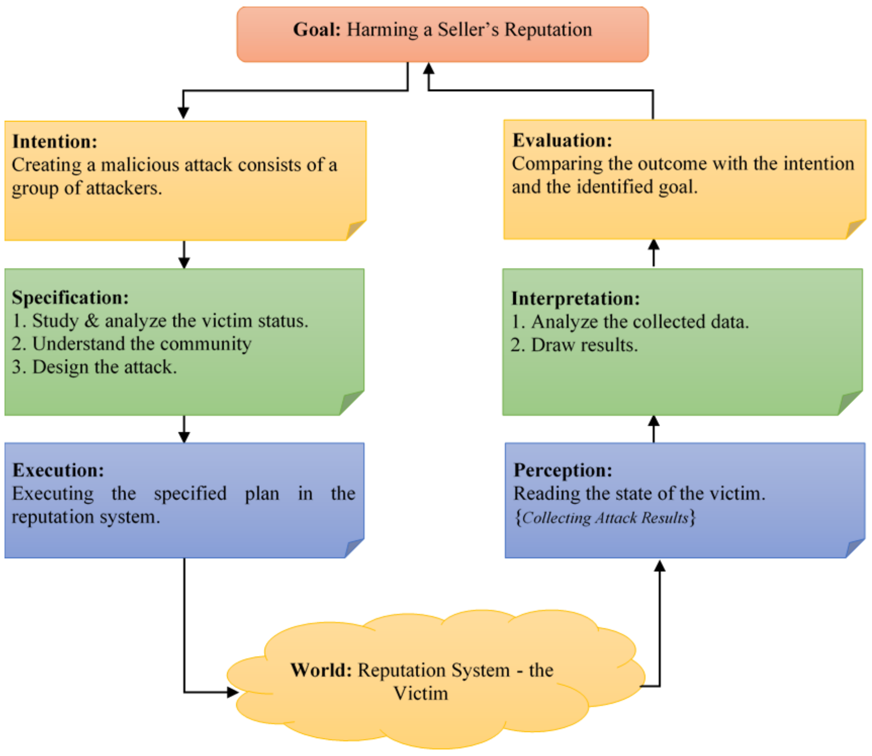

Goal: Harming a Seller's Reputation
Intention: Creating a malicious attack consists of a group of attackers.
Specification: 1. Study & analyze the victim status. 2. Understand the community 3. Design the attack.
Execution: Executing the specified plan in the reputation system.
Evaluation: Comparing the outcome with the intention and the identified goal.
Interpretation: 1. Analyze the collected data. 2. Draw results.
Perception: Reading the state of the victim. {*Collecting Attack Results*}
World: Reputation System - the Victim

Figure 8. Case Study of Applying Norman's Theory.

From this scenario, it can be concluded that Dual-system theory and Norman's theory somehow converge in depicting how humans behave. Considering this malicious attack that firmly causes bias and negatively influence the trust model, this behavior is clearly connected to System 2 in dual-system theory which means that the attack is systematic, explicit, and takes much effort.

Norman’s theory is a general theory that can occur even in spontaneous actions but with different goals.

4. BLOCKCHAIN TECHNOLOGY

4.1. Introduction

A blockchain is a cryptographically immutable, transparent, distributed, append-only ledger with a decentralized consensus mechanism [59]. Blockchain-based systems can append data to the ledger but prohibit modifying the prior data. Data is injected as records that are grouped chronologically in blocks, and each block is linked linearly with the previous block using cryptographic hashes [60, 59]. Bitcoin is the first and one of the renowned applications of blockchain introduced by Satoshi Nakamoto in 2008, and the software was released in 2009 [60]. Bitcoin is a payment network presented as a cryptocurrency that has expeditiously attracted the attention of many interested. The mechanism of blockchain helps Bitcoin to apply the concept of pseudonymous payment through generating private and public keys [61, 59]. The sellers and buyers can directly exchange over the internet in a peer-to-peer (P2P) fashion without any kind of supervision using their digital addresses. Blockchain consists of a set of protocols that allow users to transact with one another in a secure environment. Thus, transferring values from one address (account) to another on the blockchain implies that the underlying blockchain system is highly trusted. The scattered accounts do not require trust to be present between parties, but they place their trust on the mechanism of blockchain itself through consensuses that ensure the resemblance of all the copies of the distributed ledger.

The mechanism emphasizes that the processing power will be distributed equally, and the miners can precisely verify the identity of the sender, recipient, and specified amount. Due to the distributed nature of blockchain, the authority is withdrawn from a single entity and distributed to all entities in the network. In this way, Blockchain formulates a new way of interaction through enforcing rules and using some algorithms and consensus, which eliminates the necessity of checking the credentials and certifying who is trusted. Therefore, the concept of trust is established by securing the chain using specific protocols, validating the transaction, and the blocks for tamper-proofing, verifying the availability of the resources to guarantee the transaction execution [62]. All these operations and others create what is called a trust trail in the blockchain. The term "trustless" becomes one of the characteristics of blockchain-based systems that connotes trust is not needed for parties to interact with one another but only trust the blockchain. Table 3 summarizes some blockchain features.

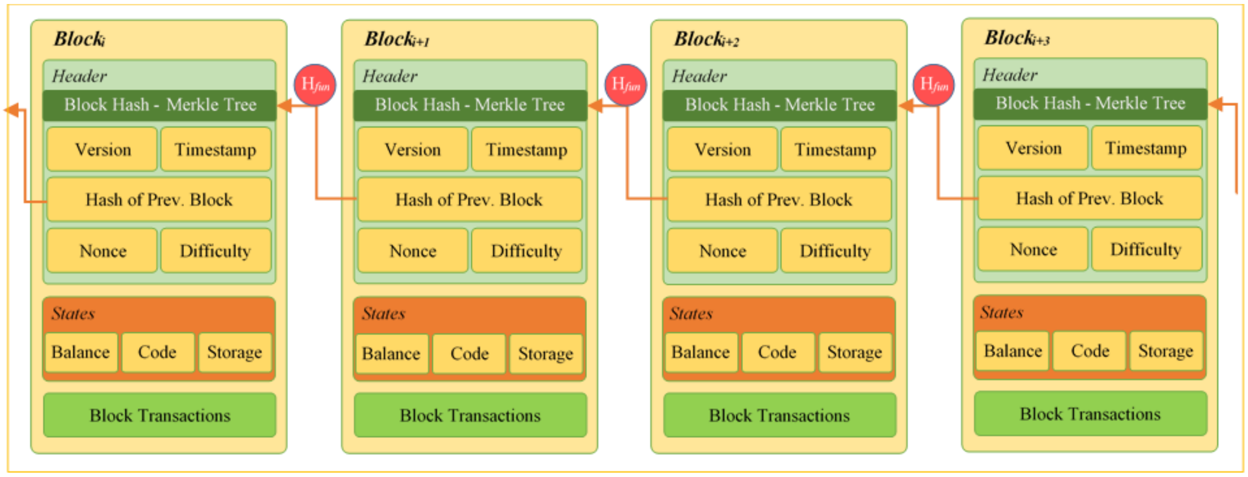


Figure 9. Simple Structure of the Blockchain System.

Table 3. A Summary of Blockchain Features.

Property	Description
Immutability	All recorded transactions on the blockchain cannot be adjusted (tamper-proof).
Decentralization	Copies of blockchain can be accessed, made, distributed over a P2P network by any node. This means any two nodes can transact with each other without intervention from a third party.
Auditability & Transparency	As public blockchains are open and accessible by any node, they can look up and verify transactions. Anyone can trace the transaction history (transparency).
Fault tolerance	the prevalence of blockchain replicas, any faults or inconsistency have occurred, it can be determined using decentralized consensus and can be cured using blockchain replicas.

More technically, the blocks are chained with timestamps and validated by miners. Elliptic curve cryptography (ECC) and SHA-256 hashing are employed to guarantee data integrity and authentication [63, 64]. As miners are in charge of verifying and validating transactions, in the bitcoin system [65, 64], the validation process computes a hash with leading zeros to meet the difficulty level. Figure 9 portraits a simple structure of the blockchain system. Every block made up of the block body that consists of a collection of transactions and the block header that holds the block own hash, a version number to track protocol upgrades, and a hash points to the previous block. The header further consists of a timestamp, the number of transactions, block size, nonce, and Merkle tree fields. The nonce is an abbreviation for number only used once. This value is added to a hashed block, and when it rehashed meets the difficulty target. The nonce number is used by the proof-of-work algorithm, and miners compete to reach this number to solve the puzzle. A difficulty level is a number to govern the time needed to create a new block. In bitcoin, every 10 minutes, a new block is added, whereas, in Ethereum, every 17.5 seconds, a new block is added [66, 67]. This value is updated periodically to make sure that the specified time is met. Merkle root is also called a hash tree. It is a hashed based data structure represented in a tree form shown in figure 10 where every leaf node is a cryptographic hash of data block, and every non-leaf node is a cryptographic hash of its sub-nodes. This mechanism is widely used in distributed systems as an efficient method for data verification [64].

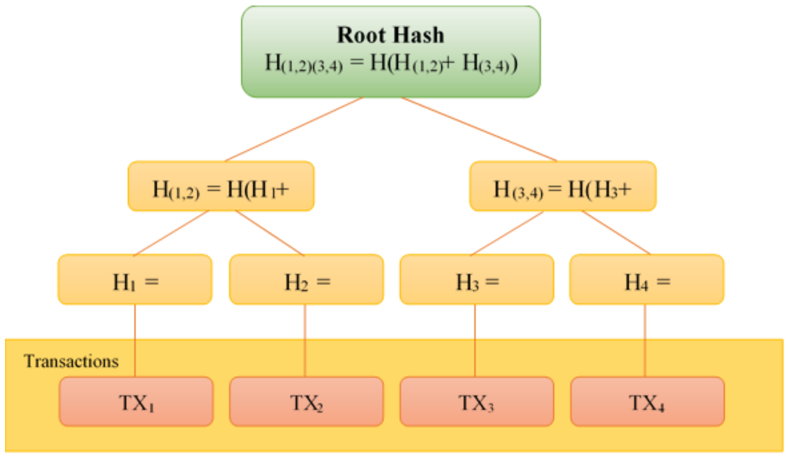


Figure 10. Merkle Tree.

4.2. Types of Blockchain

There are three major types of blockchains which are public, private, and consortium blockchains. Each type of blockchain will be succinctly elucidated in the following sections, and table 4 recaps the comparison of blockchain types.

4.2.1. Public Blockchain

It is also called a permissionless blockchain in which users can join and transact with no access restrictions. This type allows users to view and submit transactions to the public ledger. It is not necessary to join the network to submit transactions since some users can join as validators. Besides submitting and reading transactions, users can develop and submit smart contracts to the blockchain without given permission. No single transaction can be added to the blockchain before being valid. The Public blockchain is fully decentralized, which technically implies that no central authority exists to control the network [68, 69]. Additionally, the level of privacy of this kind is very high as users are not required to disclose their identity. Transparency is another attribute of the permissionless blockchain as all transactions are open and perceptible to all actors. Bitcoin, Ethereum, and Litecoin are well-known examples of public blockchains.

4.2.2. Private Blockchain

It is also called a permissioned blockchain connoting that this network is not open to the public. It is only open for a specific group to join the network and usually through an invitation from the network administrators [68, 69]. An access control mechanism is mandatory for such a network to manage users and their permissions. Another explanation is that this blockchain comes with centralized writing permissions and viewable to the public, or it can be completely restricted. Typically, every participant in the network can recognize other participants' identities. Nevertheless, transaction details are only available to participants who have the proper permission. An example of private blockchains is the Hyperledger project from the Linux Foundation. With this type, the consensus process will be more productive as all actors are not required to participate in the process as a single node owns and controls block creation.

4.2.3. Consortium Blockchain

It is also called a federated blockchain. It is a semi-public/ semi-private permissioned network in which the consensus process is controlled by predetermined nodes across multiple organizations using a collection of rules all participants agreed upon [69]. The role of the privileged nodes is to sign and validate the block before added it to the blockchain. This network falls between public and private blockchains (a combination of centralization and decentralization). The blockchain copies are exclusively distributed among eligible actors (partially decentralized). It can be said that consortium blockchain is controlled by multiple central authority (a group of approved users) contrary to the private blockchain, which is controlled by a single authority. This sort of network employs some of the cryptographic characteristics of the public blockchain with more control from the central part. This type helps to leverage information across organizations (cross-discipline) in order to enhance transparency, responsibility, and more other values.

Table 4. A Brief Comparison of Blockchain Types.

Property	Public	Private	Consortium
Consensus Mechanism	All miners	Central entity	Set of preselected actors
Immutability	Practically impossible	Can be manipulated	Can be manipulated
Centralization	Decentralized	Centralized	Semi-centralized
Consensus Process	Permissionless	Permissioned	Permissioned
Protocol Efficiency	Low	High	High
Power Consumption	High	Low	Low
Actor Identity	Pseudonymous/untrusted	Designated actors/trusted	Designated actors/trusted

4.3. Byzantine Generals Problem (BGP)

The Byzantine Generals Problem is also known as interactive consistency and Byzantine fault. It is a state when one component or more of a computer system fails to function properly. This failure may lead to delivering defective information to other components in the system, revealing a problematic behavior. The term introduced in [70] synopsizing the problem of the reliability of distributed computing systems and their solutions. The proposed strategies are employed in the development of the blockchain system to be more reliable [71]. Byzantine fault tolerance (BFT) is a term derived from BGP, and it indicates how a computer system becomes a fault-tolerant to this kind of failure [72]. One popular example of BFT is a proof-of-work consensus used in the Bitcoin system to allow the system to cope with Byzantine failure.

4.4. Smart Contract

It is also called a crypto contract, which is simply a computer program that runs on blockchain. A Smart contract manages, verifies, and regulates an agreement between parties and automatically enforces the specified obligations. The term engraved by Nick Szabo when he proposed this mechanism in the 1990s as a computerized transaction protocol that carries out the contractual terms of an agreement [107, 108]. The smart contract can be utilized not only in a form of classical contract but also in terms of a general-purpose computation. The smart contract further works as an instrument that establishes trust between untrusted parties, which eliminated the necessity of trusted third parties. Smart contracts consist of contract storage, balance, and code.

These contracts are implemented on the top of blockchains and converted into executable code. After deploying the smart contract, a 160-bit address will be allocated to this contract. This address is considered a handle of the contract that will be executed when it is used in any transaction. The smart contract can read and change the state of its storage [109]. The execution of each statement of the contract is stored in the blockchain as an immutable transaction. One major challenge is that once a smart contract placed on the blockchain, it cannot be changed or fixed in case of finding bugs. Therefore, the design of smart contracts should be done with caution to avoid such situations. From a programming perspective, smart contracts can be viewed as forms of objects as in object-oriented programming (OOP) languages (C#, Java, etc.), which means developers can use all OOP concepts when developing smart contracts with a slight difference. For instance, developers can leverage from SOLID principles (single responsibility principle, open/closed principle, Liskov substitution principle, interface segregation principle, and dependency inversion principle), design patterns, etc. All these principles help and ease the process of design DApps and reduce the need for change.

4.5. Ethereum Background

Ethereum is an open-source distributed public blockchain platform launched in July 2015 [73]. It is also featured by a state transition system in which assets can be transferred directly between accounts. The cryptocurrency of Ethereum is named Ether, which produced by the platform as a reward for the winning miner that solved the puzzle. Ether units are represented in wei, and one ether equals 1E18 wei. Ethereum is represented by two types of accounts: externally-owned accounts (EOAs) and smart contract accounts (SCAs). EOAs are governed by private keys, while SCAs are governed by their code, and these accounts have ether balance. As a programable framework, Ethereum allows developers to write their own codes through smart contracts. Smart contracts, as illustrated above, are simply pieces of code that represent units of code. These contracts can be compiled by Ethereum Virtual Machine (EVM) using the public network nodes [74, 107]. One of the popular programming languages that is used to write decentralized apps (DApps) for Ethereum is Solidity. It follows the paradigm of object-oriented programming (OOP). A “contract” in solidity represents an object that will be placed on the blockchain. Instantiating a contract into a smart contract account (SCA) can be done by performing a transaction, or a contract calls a method located in another contract. Every contract is connected to an account (SCA). The instantiated contract will be tied to a reference, which is a unique address to be used when using the contract. Additionally, multiple inheritance and polymorphism are supported in Solidity, and abstract contracts are supported as well with which the developer can apply several design principles that are compatible with OOP principles. To measure the computational work of executing transactions or smart contracts, Ethereum uses Gas as an internal pricing mechanism. Thus, transaction fees (Gas) are determined by the computational complexity, bandwidth, and storage required for the operation. EVM is hence Turing-complete as such mechanism aids in alleviating spam and resource allocation.

4.6. Trust Concept in Blockchain

Despite the powerful mechanisms of blockchain that promote the circumvention of trust, people still necessitate trust with each other, which means it cannot be removed from the equation. Trust is crucial to society and is not obsolete. It is apparent that trust in the blockchain is replaced by the aforementioned processes, which are called algorithmic trust [74], yet in fact, algorithmic trust does not represent trust between users. For example, when using bitcoin to purchase some products, there is no guarantee that the goods or services delivered by the provider will gain the desired satisfaction. Bitcoin is just a currency, and its system only ensures that the transaction is executed correctly in a safe environment. The two-way transaction is when transferring bitcoins in return for fiat/services/goods, but the Bitcoin system merely tracks one-path of the transaction (tracking bitcoins transferring) and leaves the other one without scrutiny. The system does not cover many situations and circumstances that occur outside the system such as dealing with dishonest parties. This motives to take one further step to handle such a situation where it promotes parties to behave properly within the ecosystem. Another challenge is when a face-to-face interaction takes place between two individuals who communicate online and then decide to meet each other to exchange the product in return for transferring bitcoins. The question is how the two trust each other with the absence of a system that shows the trust score for each person. Sas and Khairuddin [75] interviewed a group of bitcoin users regarding their expertise and trust challenges. They emphasize that the main concern is the risk and trust problem of dealing with dishonest users. The findings show that one-fourth of the participants were victims of fraud and misplaced their trust. Also, other participants explained the concern of dealing with anonymous people. Based on their study, the authors found that the participants prefer not to use bitcoin to buy goods but use it as a store of value due to the lack of institutional trust.

Another point connected to the trust issue is in Ethereum where developers can build their smart contract that is a piece of code portraying a collection of rules that can be injected into the Ethereum system. The contract depicts an agreement between two parties to interact with each other, and the contract is automatically executed when the predetermined rules are met in order to govern the transaction [62]. Authors in [76] argue that around 40 percent of smart contracts deployed in Ethereum is not trustless infringing the rule of non-manipulability, which means the participants cannot verify such smart contracts. Additionally, they state that trust in individuals is desired even in the case of trustless smart contracts [76]. This, in turn, again steers us to the point that monitoring the behavior of an individual is quite substantial. People who have high trust scores usually behave honestly. In other words, in the case of something wrong by others Intentionally or unintentionally, the honest people will often endeavor to solve the problem since it is linked to them. Since blockchain-based systems are P2P exchanges, trust can be established through building trust networks that can be viewed as social networks where edges represent relationships between users that may hold positive or negative weights. These edges are dynamic and evolve by carrying out interactions over time. They are also relative, as peers may trust or distrust one another at fluctuating and networked levels. For instance, user X may fundamentally distrust user Y but highly trusts user Z. There may be myriad possible levels of trust. The proposed model is built on the concept of social network, and all users are treated as nodes.

4.7. Blockchain-Based Trust Model

The growth of reporting security breaches incidents steers the attention toward an important question concerning users' privacy and ownership. Third-party authorities are responsible for collecting, storing, securing, and controlling such private information. Due to some seen/unseen vulnerabilities, these central authorities are deemed as easy targets to cybercriminals, and users may lose control of their assets/data. One of the important assets that individuals/entities may lose is their trust and reputations. The study proposes a blockchain-based model that ensures that actors can maintain their assets. This model ensures delivering some crucial values such as interoperability between different online platforms, transparency, and ownership, plus reducing the risk of cyberattacks that targets the reputation of actors. Furthermore, the proposed model introduces a solution of replacing multiple reputations on multiple different platforms by streaming these assets from centralized platforms to a decentralized platform.

Due to the increased rate of threats that ordinarily aim to purloin private and sensitive information, the importance of managing identities becomes more prominent in the virtual world. Even in the real world, one person may have a bunch of identity documents such as driver license, passport, school Id, library Id, and others. In cyberspace, online identity is globalized, and it does not recognize national boundaries. Identity management (IdM) is a framework that concerns the management of identities that are commonly connected to individuals, their authentications, authorizations, and privileges. IdM models are confined to specific domains with a commitment to be tied to these domains' boundaries [77, 78]. The aim of these models is to find an environment that is secure and effective in reducing cost, time, and repetitive tasks.

However, most of these models are centralized and monolithic. Being centralized means that these models are controlled and managed by a third-party authority and vulnerable to security breaches. The data breach has become a critical problem since it happens on a regular basis. In 2017, Yahoo suffered a significant data leakage of its users' data, which around 3 billion accounts were breached. Another incident occurred with Uber when 57 million users' private data got breached [16]. Once the leakage occurred, it is possible that the dark web gets advantages of masquerading identity, damaging reputation, and invading privacy. These incidents emphasize that users have no control and no ownership of their identities [77, 78, 79, 80].

The developed model focuses on managing identity from the angle of trust and reputation inasmuch as there is no enough work highlighting the importance of cross-community trust management using blockchain. Some identity management principles have a strong connection with managing multiple trust and reputations. Therefore, reputation management can get benefits from applying some identity management strategies and principles. In some cases, it is unfair to have multiple identities and multiple reputations for the same actor, particularly in e-commerce communities. For instance, one seller is already running a business on Amazon; this seller may decide to expand his/her business and open a new account on eBay selling goods and/or delivering services. In this case, the same seller owns multiple unconnected accounts and multiple unconnected reputations placed on different platforms. The new business on eBay will suffer the cold-start problem when the business has no previous history or reputation. Buyers may be afraid of dealing with sellers who have no reputation. The business owner will struggle at the beginning until building a good reputation. Nevertheless, the business may be collapsed before building its reputation. This seller may desire to migrate his/her good previous Amazon reputation to be used on eBay, yet it is inapplicable. All of the existed online platforms lack a mechanism to share reputation information. On top of that, actors have no control over their reputations, and they may anytime lose their ownership. Assuming that one entity has built its good reputation for years, and suddenly, the third-party platform that hosts this entity decided for whatever reasons to shut down its business/servers forever, this entity, as a result, will lose the ownership of its reputation and interaction history.

Blockchain technology can play a significant role in solving the problem of losing control and ownership, plus alleviating the impact of malicious attacks. Additionally, it can improve interoperability for managing identity/reputation because the reputation is currently managed separately at each online platform, and there is no connection between these platforms that have similar information that can improve the accuracy of trust and reputation. Also, the administrative efficiency can be ameliorated as well as enhancing accessibility. Building a blockchain-based trust model that can map all necessary information from different platforms in order to generate an accurate trust score for each individual/entity. The notion of employing blockchain establishes decentralization, which means all reputation data can be stored in a ledger where no single authority, tamper-resistant, and transparent. Moreover, sharing information among multiple online platforms helps to lessen the influence of malicious attacks that aim to manipulate actors' reputations dishonestly. The foundation of this blockchain-based trust model is the TrustMe model as an internal trust engine.

4.8. Identity Management

Identity management (IdM) falls under the umbrella of IT security. As identity is a paramount element of society, the major purpose of IdM is how to connect the rightful actor to the appropriate computerized resources. IdM indeed pertains to policies, standards, and capabilities that concerns authentication, authorization, and accountability [77, 78, 79, 80, 81, 82, 83]. These concerns are bonded to human, software, or hardware for assuring the validity of the identity attributes. IdM models are classified into three major categories: siloed, federated, and sovereign.

4.8.1. Siloed Identity Model (SI)

The most commonly used IdM models are isolated and centralized where all credentials are stored and managed by a single authority (identity provider "IdP"). It is a straightforward model where a centralized entity issues credentials to subscribers or allows them to generate their credentials to access services provided by that centralized entity. In this setup, credentials became shared secrets between the centralized enterprise and the subscriber, which is supposed to create a channel of trust between them. These credentials are ordinarily pair of username and password. In some instances, it is aggrandized to involve security questions or supplemental factors such as biometrics or physical token (multifactor authentication). Identity holders do not have control over their credentials, which makes them susceptible to attacks and misuse. Any centralized system forms a central point of attack and highly vulnerable to be compromised. Thus, it is dangerous to maintain sensitive data on such systems.

4.8.2. Federated Identity Model (FI)

The impulse behind FI is to allow several enterprises to formulate an agreement with an exclusive trusted identity provider. This model fastens multiple users' identities and attributes (federation) situated on multiple independent identity management systems [81, 82, 83, 84]. All authentication and permission functionalities are managed and controlled by a trusted identifier. Users can thereby utilize the same identity to access many online services rendered by distinct providers. The mechanism of FI supports the portability of identity information athwart the federated systems since credentials can travel on demand. This establishes what is called "Circle of Trust" among these systems. A Single-sign-on system (SSO) is an essential unit of federated IdM Model, which is responsible for access control management. SSO orchestrates the authentication process between the identity holder and the services provider by passing an authentication token [83, 84]. In the case of having one user attempts to access a permissioned service, first, the service provider checks whether the user has already been authenticated by the SSO. If the authentication is not available, the service provider will inquire about the IdP for authentication. Once the service provider receives the verification, the login will proceed. Facebook, Twitter, LinkedIn, and Google dispense SSO services that grant users the right to employ their social media credentials to log into third-party applications/services. Although this model minimizes the risk level by disseminating credentials on-demand and omitting their replicas, it requires establishing a degree of trust relationship along with a suitable agreement among participated parties. In this setting, substantial matters such as consent, ownership, privacy, and transparency remain problematic as the identity provider is still centralized and owns the data.

4.8.3. Self-Sovereign Identity Model (SSI)

The term sovereignty means individuals/entities (subscribers) have the full prerogative and ownership of their assets without any intervention from external sources. SSI model enables subscribers to manage and control their identities and personal data by themselves without relying on any intermediary. Subscribers can safeguard their privacy since they can decide which portion of their data can be disclosed and shared in any interaction. They can also hoard their identities on their own devices, where no need to entrust a third-party authority. In this setup, blockchain technology can be employed to design and build such a system.

4.9. Review of Blockchain-Based Trust Models

Distributed trust and reputation management systems are widely used in different domains. In this section, a set of proposed distributed trust and reputation models that uses blockchain will be discussed in a nutshell. These models aim to enhance some principles such as trustworthiness and privacy.

In [85] a reputation system based on the blockchain technology was proposed. Their work intends to address the issue of quantifying reputations by getting rid of personal opinion from the transaction. The model records solely a single score either one if the transaction performed satisfactorily and zero if the transaction performed unsatisfactorily. The transaction is classified positive if and only if the user received the requested file otherwise is classified unsatisfied. The transaction is carried out between the sender and requester, and the requested service is a piece of data (file) signed by the private key of the sender. In case of getting the right file, the user can send a transaction holding a reputation score, timestamp, and a hash of the received file. The miners then receive these data encrypted using the private key of the requester. To reduce malicious transactions, they suggest a proof-of-stake system (PoS) to address the problem of cold-start or users with low reputations by putting an amount of currency (bitcoins) as a security deposit.

An additional blockchain-based reputation was developed called privacy-preserving reputation system [86]. The model aims to reduce the overhead of transaction processing. A user can decide to deal with a service provider (SP) or not is based on the provider's reputation score. After performing the transaction, the user will acquire a token from SP. Thereafter, the user can evaluate the SP by passing a message made up of the gotten token, SP address, the rating score, a signature on the information, a pointer to the last review for the same SP, and optionally textual review. They employ a publish-review protocol to carry out this process. The notion behind the pointer is to rapidly to calculate the reputation as it is unnecessary reading all reputation records.

Calvaresi et al. [87] introduced a permissioned blockchain-based reputation system in multi-agent systems. The model enables agents to trace the change of their reputations after an interaction taking place. Smart contracts are used to transparently calculate the reputations of the interacted agents. The reputation scores are then stored on the blockchain along with services and their evaluations and this to secure interactions between agents. The architecture of the model provides an overall reputation score representing the average of all rating values and a specific task value of a given service and role (demander/executor). The agent first needs to be registered in the system to be able to provide and/or require services. The agent will initially get a default reputation score. Upon interaction completion, agents can evaluate one another based on the interaction outcome. The smart contract is evoked when the evaluations of both interactors received and notifying the executor with the evaluation submitted by the demander.

Authors in [88] proposed a scalable and secure blockchain-based trust management system. They try mimicking a trusted third party (TTP) through a distributed mechanism to be able to enforce policies, rules, penalties, etc. The design of the model is hybrid consists of 3-layer architecture (layer2: global, layer1: shards, & layer0: local) to minimize overhead operations and ensure scalability where access delegations and trust appraisals cab be interchanged using blockchain. The global layer concerns the security of operations among members of different clusters of layer1, which is ensured by using a public blockchain such as Ethereum and encoding access control logic in it. For every transaction executed on the blockchain, miners get paid for their computation power. The model incorporates ratings by the interacting actors, which is a component of the access delegation process fulfilled by blockchain. According to the experience-derived reputation, the decision can be made to request or delegate access to the resources or not. The trust evidence is in the figure of ratings, which can later be employed by mathematical models known as computational trust models.

5. BLOCKCHAIN-BASED TRUST MODEL “DACCT”

5.1. Introduction

The fundamental objective of trust models is to measures and supply trust information of actors of a community predicated on their behavior. There are a variety of trust models that cover several domains whether employing blockchain or not. Most of them do not target the notion of cross-community trust through sharing information. Attempts toward sharing trust information have been presented as in [89], authors propose a mechanism of common ontology to transfer reputation information between agents. Authors in [90] also propose a model for sharing reputation knowledge across online communities called CCR but using a trusted third party to manage trust information. In this setup, each online community acquires knowledge about its members, and reputation scores are calculated, maintained, and supplied to those members. Communities and their members meanwhile lose the benefit of sharing reputation information. Sharing such valuable information aids in detecting and alleviating malicious behavior. One actor has, besides, earned a reputation over time in one community, and when joining a new different community, this actor can take advantage of the portability feature of reputation.

5.2. DApp Cross-Community Trust Framework (DACCT)

We propose a blockchain-based trust model that aims to collect data from several sources to then compute and share trust information. The generated trust scores will be decentralized and stored on a blockchain platform such as Ethereum using a distributed application called DApp Cross-Community Trust framework (DACCT). We assume that communities are willing to share evaluation data and some other meta-features to produce precise and reliable trust information, and we further assume that there an identity management system has already established. The framework consists of a set of sub-ledgers and a central public ledger. Every actor owns two immutable private sub-ledgers: a ledger for given ratings and a ledger for received ratings. Also, each community owns one immutable private sub-ledger for its actors and their evaluations. The public ledger is open to all and contains trust information for all actors. DACCT provides real-time visibility of trust information for all actors.

5.3. Design of DACCT

This section presents the architecture and functionality of the DACCT that we are developing to explore the efficacy of applying blockchain technology to the trust domain. It then describes our recommendations for designing blockchain-based trust apps using familiar software patterns to address the interoperability challenges. Figure 11 shows an abstract view of the proposed framework.

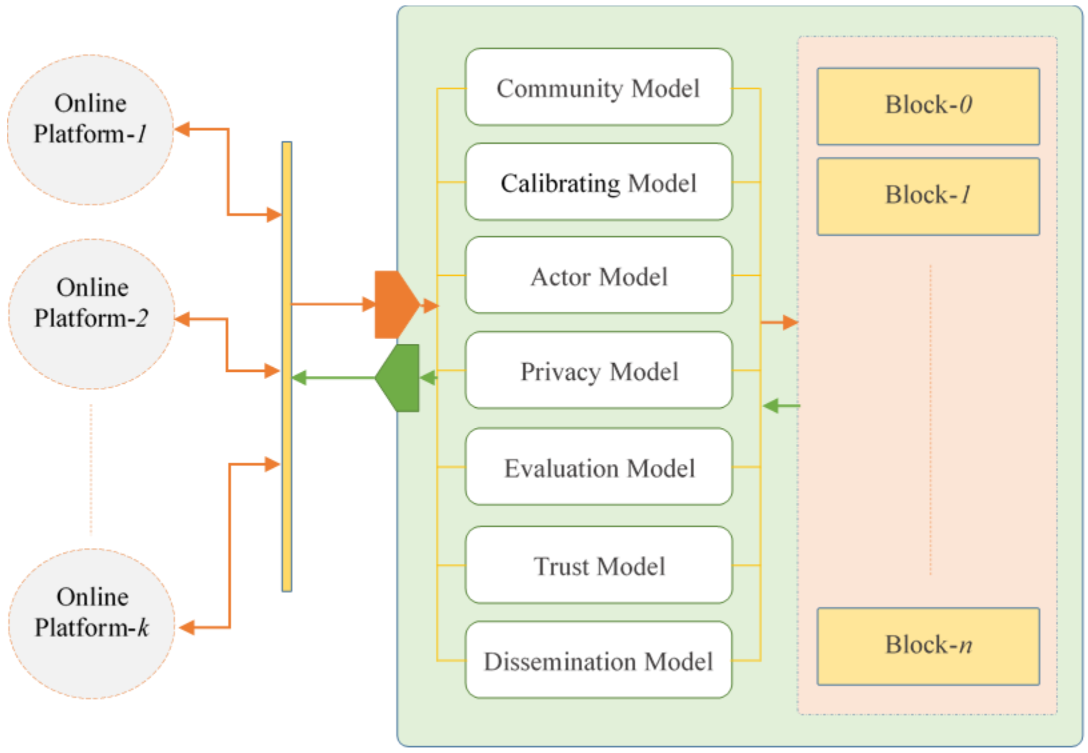


Figure 11. An Abstract View of the DACCT Framework.

The developed DACCT framework consists of a bunch of sub-ledgers. These sub-ledgers are created when an actor is registered in the system and receiving and/or giving the first evaluation. The idea behind using sub-ledger is to store only the evaluation data of a single actor which helps to rapidly retrieve the evaluation data of this actor instead of storing all actors' evaluations in one large ledger. These sub-ledgers are permissioned to the community where the transaction carried out and the assigned actor as there is a sub-ledger assigned to every actor as shown in figure 12. These ledgers are automatically permissioned to the underlying trust model to update the trust information of the involved actors. The public ledger is open and accessible to everyone and used to store trust information. This public ledger can be built using the Ethereum platform. Moreover, the ownership feature is guaranteed in two ways: first, by storing the trust information on a public ledger and secondly by allowing actors to export their ledgers and store them on their local devices. These locally stored ledgers can be verified by using its data, and if someone tried to tamper these ledgers, they will be broken and become invalid. In this scheme, a supervisor node is also needed to manage the community membership processes. This node is one of the community nodes, and it is dynamic changing periodically which means there is no persistent monitor node. The architecture of DACCT shown in figure 13 and is simply made up of three layers as follows:

- *Presentation layer:* this layer concerns the user interface of the underlying system. It allows users (actors/communities) to interact with the system. The interface could be graphical for actors and non-graphical for communities in a form of APIs.
- *Business logic layer:* it includes the core functionalities of DACCT. The intention of this layer is to process, manage, and transform data received from the data layer and applies specified rules and policies to assure consistency and validity. In our

case, it works as an intermediary that connects the presentation layer with the data layer.

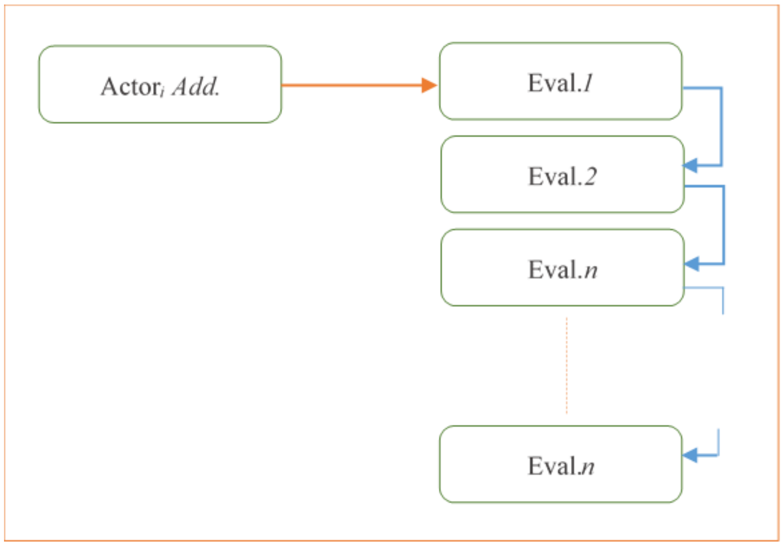


Figure 12. An Actor *i* and its Sub-Ledger of Evaluations.

- *Data layer:* it contains functionality for managing the data such as retrieving data from the blockchain and creating new trust transactions. This layer is shared by the business layer, and it works as a connection between the business logic and the blockchain.

5.4. DACCT Models

This section describes the purpose and functionality of each sub model in DACCT. These models collaborate together to make DACCT more reliable and deliver the required service.

5.4.1. Community Model

This model concerns managing community operations such as a new community joining DACCT. The new community should register with its blockchain address. In this case, we create a public ledger using Ethereum to record all communities with their nonconfidential information such as the community size attribute. This is to retain such information to be available to the public and all actors just in case one community decided for whatever reason to shut down, so the community public ledger still available and the address used on other records still available as well.

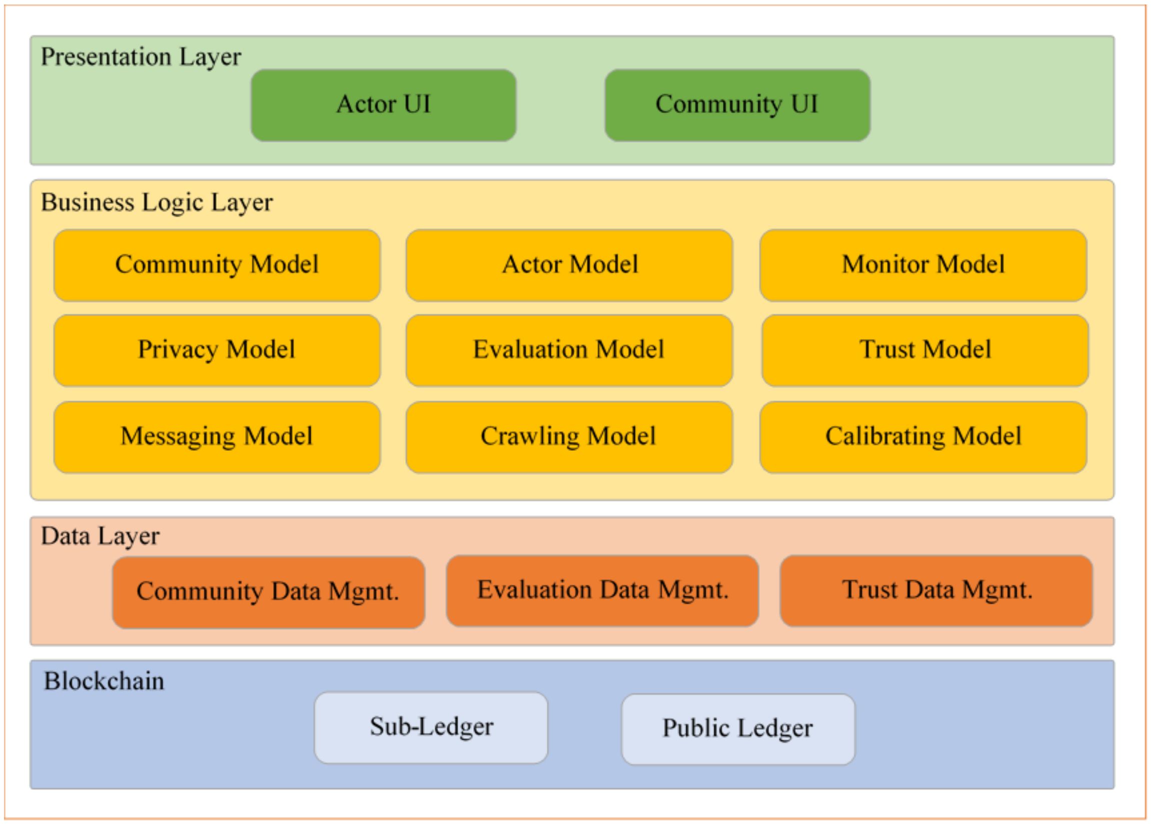


Figure 13. DACCT Layered Architecture.

5.4.2. Monitor Model

This model concerns observing the behavior of all joined communities. This model measures the confidence level of each community. When one community confidence level goes below the prescribed threshold, this model alarms all other communities informing them that this community is suspicious. The equipped measure in this model is very simple, and it can be a target of future work. If one community provides submits evaluations to DACCT with a noticeable deviation, this community should be subject to more investigation. This gives a sign of low confidence level. The variance is calculated using abnormal deviation from the mean (RDMA) computed as follows:

$$d_{ij} = \left| e_{ij} - \underbrace{Avg}_{k} \; e_{kj} \right|$$

5.4.3. Actor Model

This model manages actors' operations as communities store actors' information including the actors' blockchain addresses on each actor's private ledger. This helps to map out actors to their evaluation and the communities that submitted these evaluations. So, when we need to store the actor data on the public ledger, we can easily get the related data from the sub-ledgers.

5.4.4. Crawling Model

The function of this model is to compute factors that are used in our trust model (TrustMe) and need to be updated frequently. When a new evaluation is given, we need to recompute positive and negative medians. The popularity factor is an additional element that requires to be updated after providing a new evaluation. The computed values will be temporarily saved until a new evaluation injected to be replaced by newly generated ones.

5.4.5. Evaluation Model

This model serves as a manager of the evaluation process such as the rating system. Every community can use this model to store the evaluations provided by its users, and these evaluations are exclusively permissioned to the community where the evaluation provided and the participated actors.

5.4.6. Trust Model

The duty of this model is computing, generating, and updating trust scores based on a set of factors. The trust model that is employed in DACCT is TrustMe. The trust score will be recorded on public ledger using Ethereum to be available to the public and owned by its actors. When a new evaluation provided a new transaction will be carried out on the public ledger to post the new trust score and to end up with a new block added to the blockchain, and this is done by the dissemination model.

5.4.7. Privacy Model

This model enforces privacy policies on specified fields as a protection mechanism for sensitive data. The volume of a transaction, for example, is usually deemed private. The privacy model uses sub-models to hash or encrypt some specified values. The encrypted values are revealed and decrypted to only the permissioned participants. Additionally, the privacy model ensures unlinkability if an actor refuses to link its accounts/identities across communities even using its pseudonymous. Thus, the scans the last actor record to check whether the actor allows linkability or not.

5.4.8. Calibrating Model

It can also be called a standardization model. It operates as an intermediary instrument to make sure that all provided data are standardized and consistent. This is because communities may use different evaluation mechanisms such as the *5-star* system and [-10, +10] ratings. Thus, we crave to map out all data to its equivalence to be consistent and reliable when computing trust. Another point is that the popularity factor is generated based on the population of each community which is considered not fair if these communities vary in their sizes. For example, the following naive equation standardizes the popularity value for an actor in a community (C_v).

$$p_{a_i} = \frac{p_{a_i}}{\left(\sum_{k=1}^{n} \mathfrak{S}_{c_k}\right) - \mathfrak{S}_{c_v}}$$

where p_{a_i}denotes the popularity value of actor *i*; $\mathfrak{S}_{c_k}$denotes the size of community *k*, while $\mathfrak{S}_{c_v}$denotes the size of community *v*.

5.4.9. Dissemination Model

This model concerns propagating trust information. After computing trust scores by the trust model, these scores are passed to the dissemination model to record the new update on the public ledger as well as informing all community partners about the new update. Since trust information is public, anyone can submit a query to retrieve the trust information of an actor which is done by the messaging model. This process can also be done without using DACCT but by directly retrieving information from the public ledger using their code/DApp.

5.4.10. Messaging Model

The function of the messaging model is to exchange information between the application and external entities. It encapsulates messages with its necessary data to be sent to the target. Also, unwrapping the received message and the encrypted parts will be passed to the encryption model to decrypt it.

5.5. Software Engineering Considerations in Designing DACCT

Designing an application that is based on blockchain platforms requires paying more attention to some blockchain restrictions such as in Ethereum. The goal of this section is to design our proposed trust model to be established on a blockchain, and it aims to focus on applying some software design perspectives such as SOLD principles and some design patterns to build an effective blockchain-based application and tackle some blockchain restrictions. Software design usually is driven by quality attributes that enforce the design to go into a specific direction/solution. Since a blockchain-based system offers some values and quality attributes such as transparency, interoperability, the design should take these elements into account.

5.5.1. Maintaining Extensibility

Developers are familiar with writing code for an application that interacts with mutable data storage. With public blockchains, the situation is completely different as its data is immutable once recorded. Therefore, the design of the application shall take into account the issue of evolvability as smart contracts cannot be changed once written on Ethereum. This can be fulfilled by applying a design pattern that guarantees loose coupling and minimizes integration complexity for the client that interacts with the blockchain. Figure 14 shows the Abstract Factor pattern handling actors' accounts.

This design helps to create accounts for different types of actors such as individual actors and institutional actors. The abstract factory design also allows adding a new factory for a new type of actor that is not presented at the time of designing and building the application. The new code at least adheres to some SOLID principles such as "*open to add/close to change*" which means the existed code will not be changed which also adheres to blockchain restrictions, and the design allows the new code to be injected without impacting the existed contracts. The process of managing actors' accounts is transferred to the abstract factory. Concrete smart contracts (actor factories: *IndividualActorFactory* & *EntityActorFactory*) inherits from the abstract factory and customize its methods based on its criteria. We designed two distinct factories to apply the principle of separation of concerns (SoC) as the two actors may differ in the behavior of their functionalities. Thus, *EntityActorFactory* targets actors who represent business organizations, whilst *IndividualActorFactory* targets individuals. They are

5.5.2. Maintaining Privacy

Blockchain technology applies the notion of pseudonymous by employing private and public keys, so this somehow advocates privacy. Pseudonymity makes users use their address instead of their real names. However, this can be seen as data protection but not real privacy. However, in the case of using hybrid blockchain, actors' identities are known in advance from all joined communities. In this case, we need a mechanism or design that assures preserving privacy, and only permissioned participants can view the private data and reduce transparency level.

Another point is that maintaining transparency is high when using the Ethereum platform. Thus, we need to encrypt some fields using the public key and to be decrypted to actors who own the corresponding private key. Other public actors can only read the data stored in the plain format. In DACCT, the transaction volume and other values will be encrypted, and thus will not be revealed to the public. As DACCT is interoperable, we need to protect actors' privacy in addition to encryption. In this context, the Proxy design pattern as shown in figure 15 can be utilized to ensure preserving private data. A Proxy design pattern provides a contract interface (*ITrustData*) to other contract objects by creating a proxifier as a wrapper contract (*TrustProxy*). This proxifier may contain additional functionalities to the contract of interest (*TrustData*) without changing the code of the contract.

The considerations mentioned above are some of the design challenges that need to be addressed when building blockchain-oriented applications. The environment of the application is characterized by being immutable, decentralized, tamper-proof, and transparent. Thus, the aid of the proposed design is to improve maintainability, reduce complexity, ensure extensibility, improve integration, and in some cases, ensure privacy in such environment.

5.6. Data Structure and Message Types

In this section, we introduce some major data structures for DACCT along with some implementations in order to clarify the process and establish the foundation of developing DACCT.

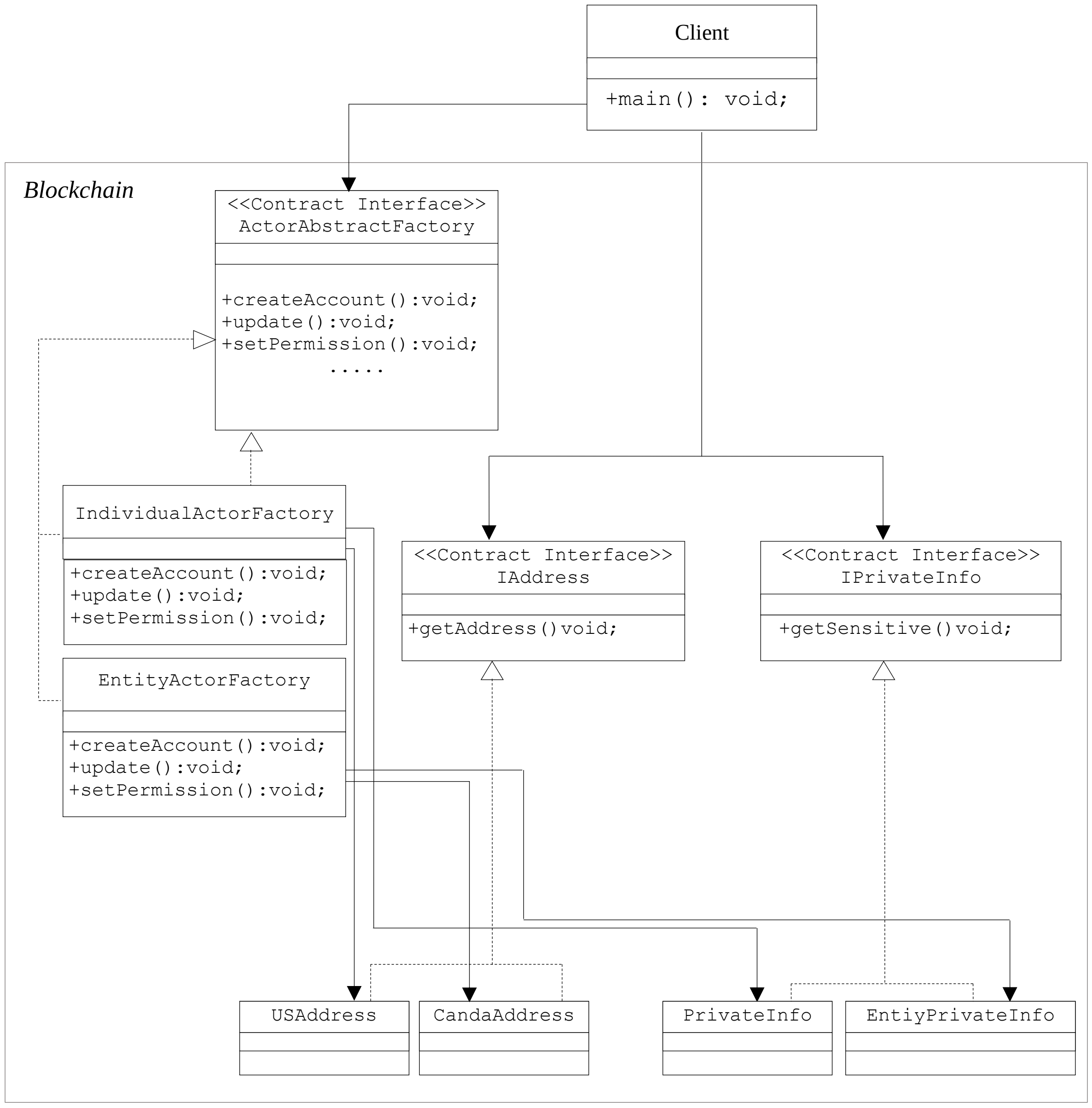
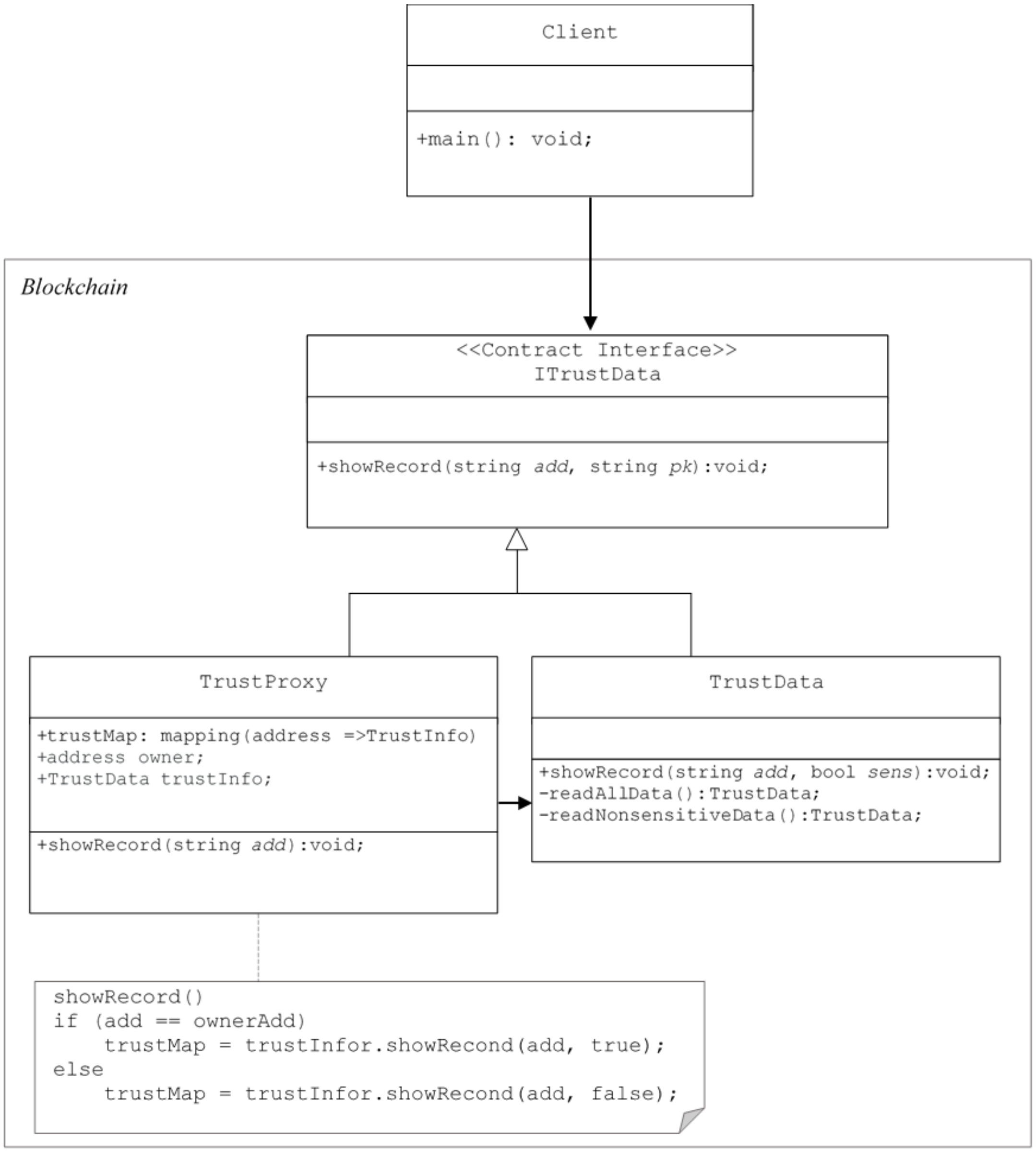


Figure 14. Abstract Factor Design Pattern for Actor Account.

5.6.1. Evaluation Record (*Eval*)

The *Eval* record represents the structure of a given evaluation in the form of ratings shown in figure 16. This record will be stored as a transaction on a block and then linked to the blockchain after validation and verification processes. *RaterPubKey* a byte array records the public key of the rater, whilst *RateePubKey* records the public key of the ratee. *RaterAdd* and *RateeAdd* are also byte arrays that store the addresses of the rater and ratee consecutively. *Timestamp*, a string filed, holds the time when the evaluation is given. *AssoCommunityAdd* is a byte array to represent the address of the associated community where the transaction performed. RatingScore, an integer value, represents the given rating, whereas *Review* is a string filed to store the comments given by the rater. *TransactionAdd* denotes the transaction that is carried out on *AssoCommunityAdd*, and the evaluation was given for this transaction.


Client
+main(): void;
Blockchain
<<Contract Interface>>
ITrustData
+showRecord(string add, string pk):void;
TrustProxy
+trustMap: mapping(address =>TrustInfo)
+address owner;
+TrustData trustInfo;
+showRecord(string add):void;
TrustData
+showRecord(string add, bool sens):void;
-readAllData():TrustData;
-readNonsensitiveData():TrustData;

```
showRecord()
if (add == ownerAdd)
    trustMap = trustInfor.showRecond(add, true);
else
    trustMap = trustInfor.showRecond(add, false);
```

Figure 15. Proxy Design Pattern for Preserving Privacy.

```
type Eval struct {
    RaterPubKey []byte
    RateePubKey []byte
    RaterAdd address
    RateeAdd address
    Timestamp string
    AssoCommunityAdd address
    RatingScore int
    Review string
    TransactionAdd address
}
```

Figure 16. The Structure of the *Eval* Record.

5.6.2. TrustInfo Record (Trust)

The *TrustInfo* record indicates the structure of a computed trust along with its attributes shown in figure 17. *TrustInfo* record will be stored as a new transaction on a block and then linked to the trail of trust on the blockchain. This record consists of thirteen fields. *ActorPubKey*, a byte array to hold the public key of the actor that this record is connected to it. *ActorAdd*, an address field, denotes the address of the actor, and it can be used as an index to reach the trust score quickly. *Timestamp*, a string, holds the time when the trust score computed, and it is usually similar to the time of the evaluation that causes recomputing the trust. *TrustScore*, a double field, represents the computed trust score. *Popularity* is a double field that holds the PageRank value of the actor. *TimeWeight* is a ratio that represents the weight of the time that is used in the trust calculation. *PosEvalCounter* is an unsigned integer to hold the total number of positive given evaluations. *NegEvalCounter* is an unsigned integer to hold the total number of negative given evaluations. *PosMed* denotes the positive median, whereas *NegMed* denotes the negative median of the actor.

```
type TrustInfo struct {
    ActorPubKey []byte
    ActorAdd address
    Timestamp string
    TrustScore double
    Popularity double
    TimeWeight double
    PosEvalCounter uint
    NegEvalCounter uint
    PosMed double
    NegMed double
}
```

Figure 17. The Structure of the *TrustInfo* Record.

5.6.3. Message Types

This section describes the messages and their contents that can be passed and received by the DACCT components and also for external calls.

5.6.3.1. Request Trust Information "req.trustInfo"

The simple request form consists of the address of the actor under inquiry, the message timestamp, the message request header ("*req.trustInfo*"), details field to retrieve only trust score or the whole trust record, and the requester address. This type of message seeks the trust score of an actor using DACCT. The message will be received by *Message Model* and after unwrapping the message, the *Trust Model* will read the recent trust score using the actor address.

message = {msg:"req.trustInfo", msgTimestamp, actorAdd, requesterAdd, details}

5.6.3.2. Respond Trust Information "res.trustInfo"

This message provides a reply to a prior trust information request. This message consists of the message header, the time when respond submitted, the actor address under questioning, the requester address, and the trust information. In case of receiving an undetailed request, the response message will only hold the trust score. While in the case of receiving a detailed request, the response message will hold all allowed fields in the form of an array.

message = {msg:"res.dtrustInfo", msgTimestamp, actorAdd, requesterAdd, trustScore} undetailed response

message = {msg:"res.udtrustInfo", msgTimestamp, actorAdd, requesterAdd, trustInfo[]} detailed response

5.6.3.3. Update Public Trust Information "update.trustInfo"

This message concerns updating the trust score on the public ledger on the Ethereum platform. This is made by invoking a *updaePublicTrust()* method hosted by the dissemination model which allows DACCT to create a new transaction. The message consists of evaluation and trust information because this record is a property owned by the actor. If one community does not exist anymore, the trust information along with all necessary information is still preserved on the public ledger. The message includes the message header, the message timestamp, the rater address, the ratee address, the computed trust score, and timestamp when the trust generated, the weight of the time of the evaluation/trust, the community address where the evaluation is given, the evaluation value, the popularity of the rater to be used as a weight for its rating value, the positive median the negative median, the total number of positive ratings, the total number of negative ratings, the transaction volume for which the rating is given, the transaction weight, and the review.

message = {msg:"update.trustInfo", msgTimestamp, raterAdd, rateeAdd, trustScore, trustTimestamp, timeWeight, communityAdd, evalScore, raterPopularity, posMed, negMed, posEvalCounter, negEvalCounter, trasnsVol, transWeight, review}

The design of DACCT is more complicated, and it is still under development. The discussed design considerations with messages and data structures just to provide a simple idea about the application design without complexity. Trust management can be implemented in different ways endeavoring to maintain particular values. For instance, ignoring the privacy issue may easy the design and implementation process. Also, if we only consider building the complete application using Ethereum blockchain, there is a crucial desire to establish a new consensus mechanism to be completely decentralized community management. This consensus concerns monitoring the behavior of communities in the case of providing misleading information and concerns the membership processes. This work will be done in the future to introduce an effective mechanism that can be helpful for cross-community-oriented applications.

6. EXPERIMENTS AND RESULTS: DENSITY-BASED CLUSTERING AND GRAPH ANALYSIS

6.1. Tools and Languages

This section summarizes the tools and languages used in the following experiments. Weka was used to apply density-based clustering. R was used for Elbow analysis, generating data distributions, Silhouette analysis. R was further employed for the analysis of Spearman's correlations and some graph operations. Gephi was also used for computing graph metrics and plotting the network.

6.2. Dataset Information

Table 5 shows the statistics of the bitcoin-OTC and bitcoin-Alpha datasets [91, 92] used in this paper. The datasets are represented as a directed graph where users are associated with nodes and the relationship between users is represented by weighted edges. Each edge holds a weight that shows the judgment between two users (*u, v*). The weights are scaled between (-10) and (+10). According to OTC's guidelines, a rating of +10 represents a well-trusted user, while -10 represents impostors. The other rating values have relative intermediate interpretations. The structure of the dataset consists of Source (node id of source, i.e., rater); Target (node id of a target, i.e., ratee.); Rating (the source's rating for the target, ranging from -10 to +10 in steps of 1); and Time (the time of the rating, measured as seconds).

Table 5. Summary Statistic of Datasets.

Dataset	Description	Number of	Range
Bitcoin-OTC	Nodes/Users	5,881	[1, 5881]
	Edges/ Evaluation	35,592	[-10, 10]
Bitcoin-Alpha	Nodes/Users	3,783	[1, 3783]
	Edges/Evaluation	24,186	[-10, 10]

6.3. Pre-Processing/Data Preparation

In order to test the models, some unavailable data is needed to be used by its corresponding factors. Transaction volume is not unavailable in the original datasets. Therefore, the values for this attribute are generated using random function. Since the two given datasets are collected from two different bitcoin platforms, the transaction volume represents the number of bitcoins transacted. In the experiment, bitcoins are converted into the corresponding values in dollars. To form the boundaries of the transaction volume, the maximum number of bitcoins is demanded to be identified. By surveying several cryptocurrency platforms (SpectroCoin, Bitso, itBit, and Coinbase), the findings show that the maximum number of bitcoins is unbounded, but the majority of these platforms supplies around 500 BTC in average as the maximum number of bitcoins. However, following the policies of SpectroCoin, transactions less than or equal to 5 BTC are performed immediately without more inspection, whereas, for security purposes, values bigger than 5 BTC are processed manually. Thus, 5 BTC is chosen as the highest peak of the transaction volume. Then, to convert bitcoins into dollars, the price of bitcoin on December 17th, 2018, was \$3, 255.37. Thus, the maximum amount is 16,276.85 dollars. The generation process used one as the lowest peak and 16, 276.85 as the highest peak, and started generating random values using this boundary. The next operation was translating these volumes into weights by considering the weight of the maximum amount as 1.0. The following equation is used to generate the weights.

$$tvw(u_i) = \frac{tv(u_i)}{max\ (TV)}$$

where: $tvw(u_i)$ denotes the transaction volume weight of user u_i; $tv(u_i)$ denotes the transaction volume of user u_i; TV represents all transactions.

6.4. Descriptive Statistics and Analysis

TrustMe model measures the reputation of users by filtering outlier values, and possibly reducing the effect of a malicious user. Out of eight introduced factors, six factors were injected into our model and primarily tested by an unsupervised approach. The main six parameters (number of ratings, positive received ratings (*PRR*), negative received ratings (*NRR*), rating age, and popularity) constitute the factors (independent variables). In the context of binary evaluation, trustworthy and untrustworthy are the outcomes (dependent variables). The dependent variables could be represented in levels using the fuzzy logic concept. The bitcoin-Alpha dataset consists of 3,783 complete users and 24,186 interactions, and bitcoin-OTC consists of 5,881 users and 35,593 interactions. The observations were assumed to be complete with no missing values.

The brief descriptive statistics for the dataset are reported in table 5. The modeling phase was started after building the model with its stimulating factors and preparing the dataset. In the experiment, a density-based clustering method with k-means as a wrapped algorithm was applied. A suite of machine learning software in the WEKA were used for the experiments. The experiments used the bitcoin-Alpha and bitcoin-OTC datasets and were validated using several k values. Initially, the trust score was computed by employing three separate metrics: one by using the raw mean, second by using PeerTrust, and thirdly, by using Trust-ME. The generated trust scores from the three metrics were clustered separately using density-based clustering approaches with several k values. The purpose of using three metrics was to make a comparison between the simple metrics (raw mean), PeerTrust (parameterized metric), and our metrics.

After finding the trust score for each user, the final step was to normalize the produced values to be clustered in the next phase. The elbow method was used before clustering to determine the appropriate number of clusters. The elbow method suggested that *k = 3, 4, or 5* were the optimal number of clusters for raw mean model and both datasets, *k = 4, 5 or 6* for the PeerTrust model, and *k = 4, 5, or 6* for TrustMe model. However, during the experiment, the number of clusters used was *k = 2* and *k = 5* to show how the distribution of the data would look like and to further illustrate several levels of trust. Number five has been chosen because the Elbow function shows that all models include *k = 5* as a possible optimal number of clusters. The models can be used to represent the trust scores in a binary form as trusted or untrusted, or by using a fuzzy logic concept to represent trust in several levels such as highly trusted, trusted, semi-trusted, barely trusted, untrusted, and highly untrusted. Figures 18-23 show the charts that the Elbow method has drawn using the trust scores produced for all trust models and the two datasets.

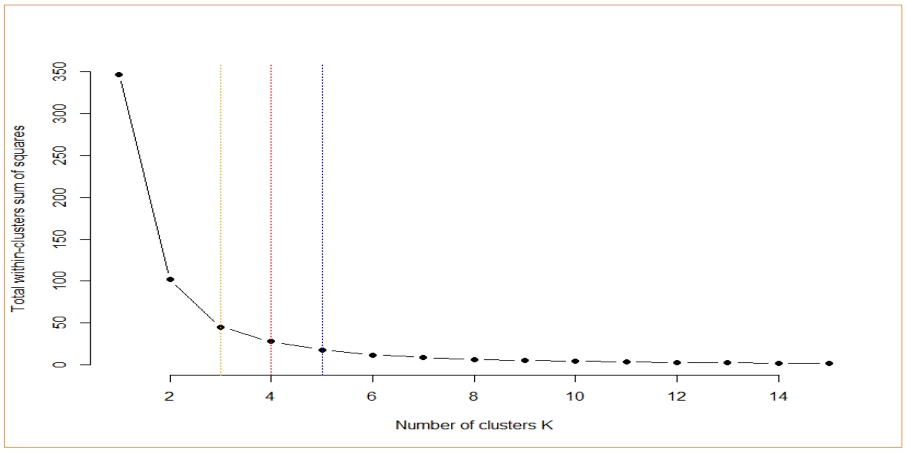


Figure 18. Elbow Method - Optimal Number of Clusters for RawMean of Bitcoin-OTC.

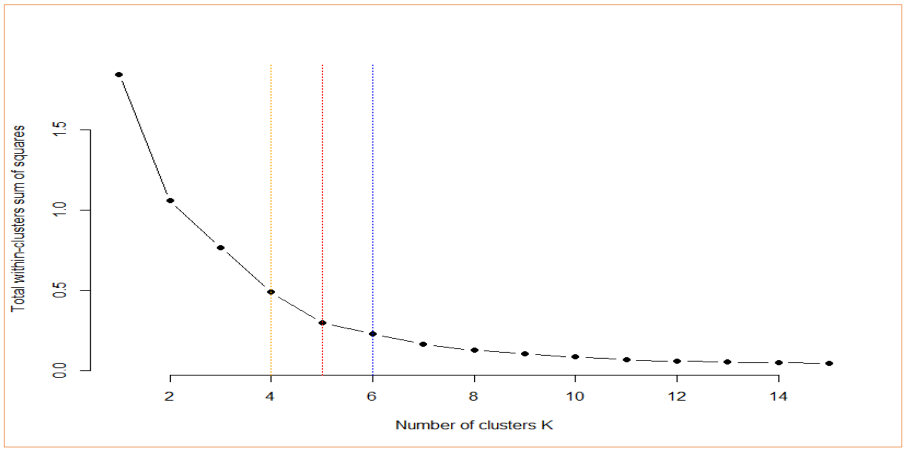


Figure 19. Elbow Method - Optimal Number of Clusters for PeerTrust of Bitcoin-OTC.

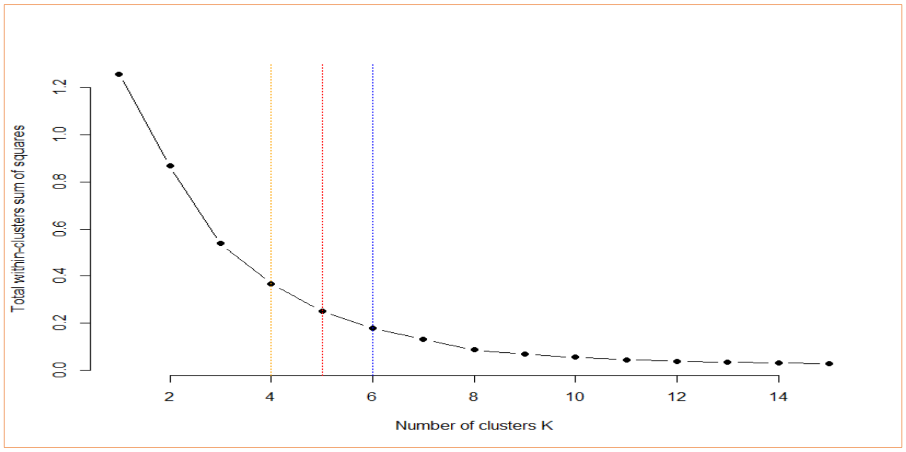


Figure 20. Elbow Method - Optimal Number of Clusters for TrustMe of Bitcoin-OTC.

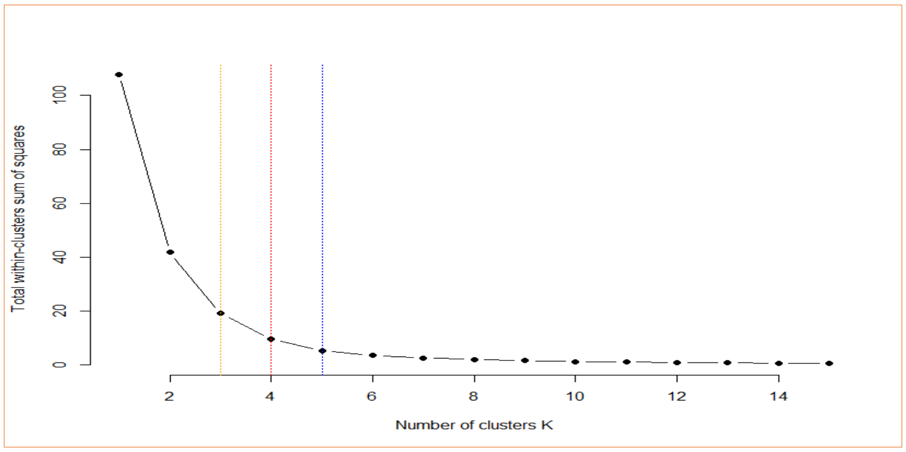


Figure 21. Elbow Method - Optimal Number of Clusters for RawMean of Bitcoin-Alpha.

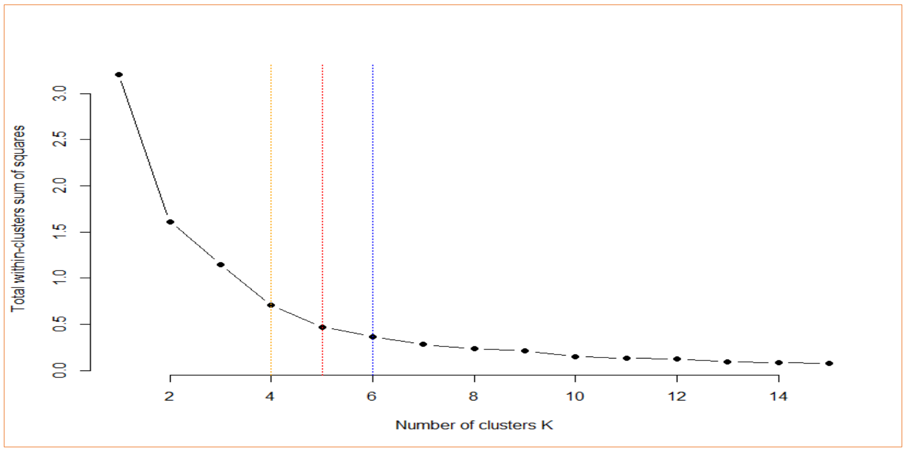


Figure 22. Elbow Method - Optimal Number of Clusters for PeerTrust of Bitcoin-Alpha.

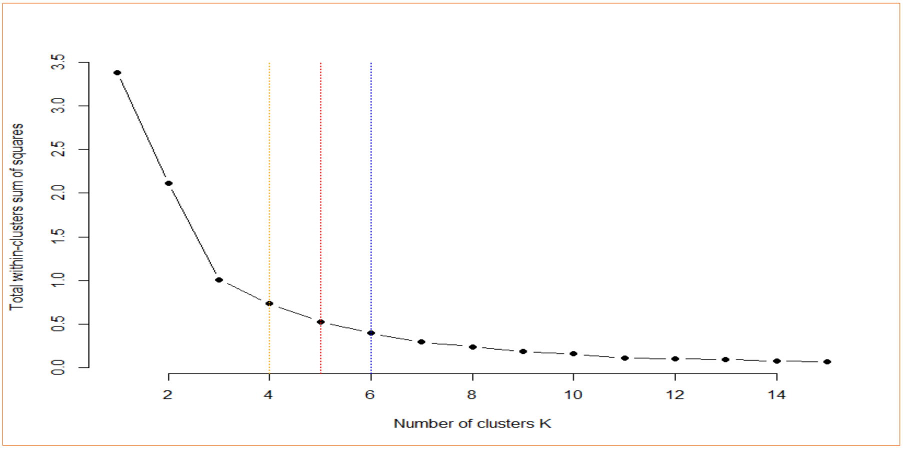
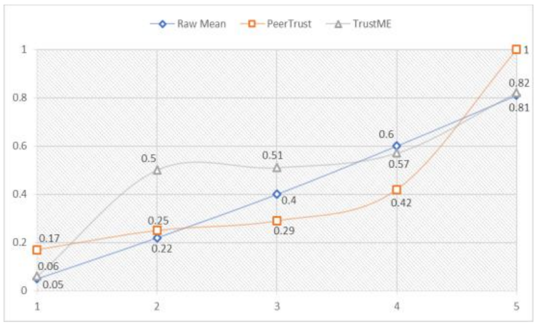


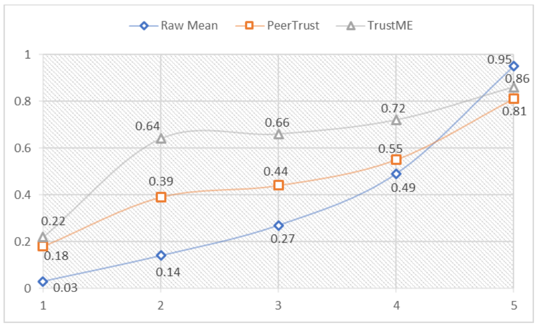
Figure 23. Elbow Method - Optimal Number of Clusters for TrustMe of Bitcoin-Alpha.

Commonly, classifying users into different categories of trust requires establishing thresholds. The threshold is a dispositional property that is hardly understood and depends on the personality and the identity of an individual. Uncharitable people/users may enforce a higher threshold, while a lower threshold could be used by lenient users [93].

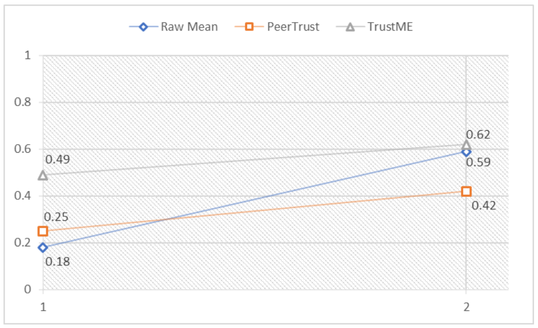
Based on the observation of existing works, clustering is an unsupervised approach is rarely used in evaluating trust models due to the difficulties of its validation process. Nevertheless, clustering could be quite useful in evaluating trust models by treating cluster centroids as thresholds and placing users based upon their trust scores to the closest threshold. For instance, the applied clustering method uses the k-means algorithm in the background where data points (trust scores) are divided into *k* number of groups/clusters. This process should guarantee that intra-cluster similarity is high and inter-cluster similarity is low. Figure 24 shows the computed centroid to the generated clusters for all the two datasets.



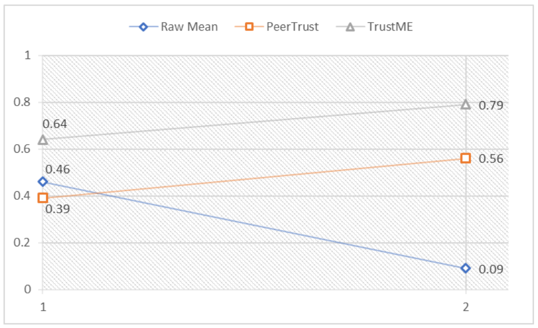
OTC-Bitcoin Centroids - Five Clusters



Bitcoin-Alpha Centroids – Five Clusters



OTC-Bitcoin Centroids - Two Clusters



Bitcoin-Alpha Centroids – Two Clusters

Figure 24. Clusters Centroids for All Models using Bitcoin-Alpha and Bitcoin-OTC.

6.4.1. Euclidean Distance

Mainly, the mean value of data points in a cluster is used to measure the similarity. In our experiment, the Euclidean distance function was exercised to measure the similarity between the trust scores and the cluster centroids, and this process continues until arriving at the convergence criteria. Minimizing the value of the square-error function is demandable, and it is calculated by aggregating the Euclidean distances between each data point and its cluster centroid.

With minimum square-error, it is possible to obtain clusters in the shape of compact hyper-ellipsoids where the clusters are well-divided from one another. As a result, the separation of trust scores would be more accurate and acceptable since each value is placed in a suitable category/cluster. However, in the real world, trust scores are irregularly distributed. Therefore, employing a density-based approach is more appropriate for this type of data instead of utilizing a simple k-means approach. Figure 25 shows the distribution of the computed data (trust scores) for the three models using Bitcoin-OTC and Bitcoin-Alpha datasets.

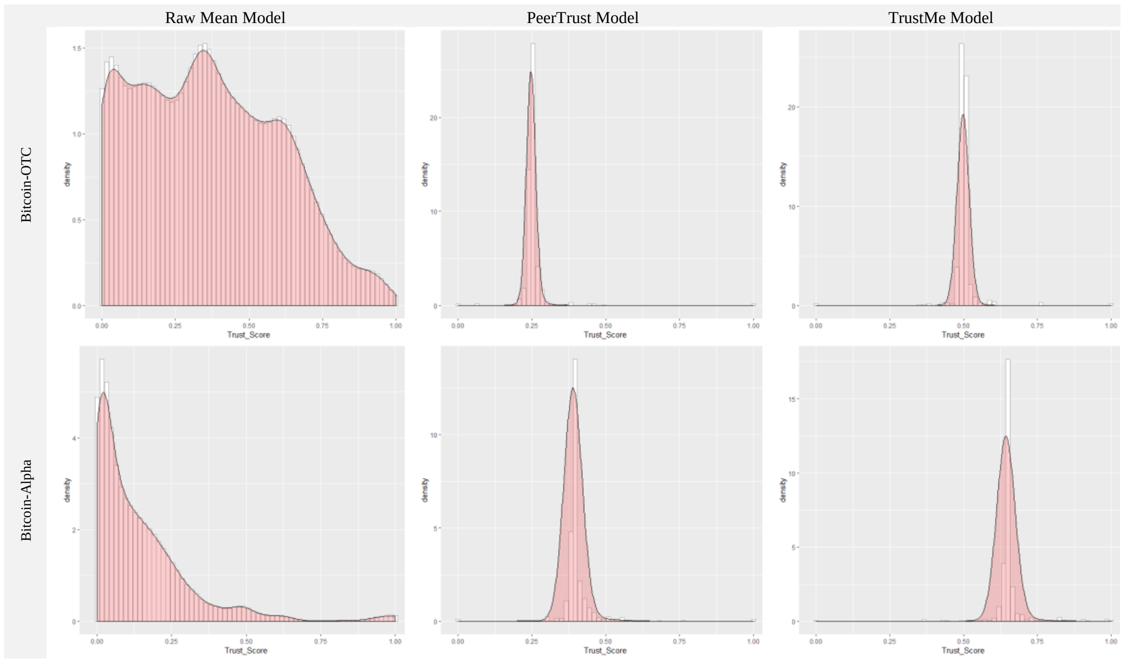


Figure 25. Data Distribution – Histograms and Density Curve for Trust Scores.

6.4.2. Clustering and Sum-Squared-Errors Analysis

Table 6 and Figure 26 show the clustering results by generating two clusters for RawMean, PeerTrust, and TrustMe models using the two datasets. The first cluster is labeled as trusted for the highest mean, whereas the second cluster is labeled as distrusted for the lowest mean. The table shows the population assigned to each cluster based on the final centroid. In this binary evaluation, Table 6 shows that the number of iterations for RawMean model is 18 within each cluster sum of squared errors around 41.74 using bitcoin-Alpha, and 10 iterations within a sum of squared errors = 101.82 when using bitcoin-OTC, which is considered very high due to the farness among trust scores. The number of iterations for PeerTrust is 12 within SSE = 1.61 for bitcoin-Alpha, and 15 iterations within SSE = 1.06 when using bitcoin-OTC. Finally, the number of iterations for TrustMe is 11 within SSE = 2.27 for bitcoin-Alpha, and 14 within SSE = 0.93 when using bitcoin-

OTC. By observing these results including the results of multiple evaluations (five clusters), it is noticeable that PeerTrust and TrustMe compete against each other based upon the clustering validation. The number of iterations and the sum of squared errors are small for both models and close to each other, and this due to the closeness among the generated trust scores when using PeerTrust and TrustMe models. Table 7 and figure 27 show the results of generating five clusters for all models.

Table 6. Statistics for Two Clusters of the Three Trust Models.

Data	Model	Cluster	Initial Centroids	Final Centroids	Prior Probability	Normal Dist.		# of Instances	Percentage	Cluster Label
						Mean	*Std. Dev.*			
Bitcoin-Alpha	Raw Mean	0	0.159949	0.4616	0.1456	0.4616	0.1972	514	14%	Trusted
		1	0.140666	0.0867	0.8544	0.0867	0.0793	3269	86%	Distrusted
	PeerTrust	0	0.38983	0.5637	0.0143	0.5637	0.1081	77	2%	Trusted
		1	0.385604	0.389	0.9857	0.389	0.0163	3706	98%	Distrusted
	TrustMe	0	0.649086	0.7885	0.0143	0.7885	0.0725	48	2%	Trusted
		1	0.639352	0.6425	0.9857	0.6425	0.0231	3735	98%	Distrusted
Bitcoin-OTC	Raw Mean	0	0.159678	0.1805	0.5691	0.1805	0.1246	3374	57%	Distrusted
		1	0.805003	0.5928	0.4309	0.5928	0.1403	2507	43%	Trusted
	PeerTrust	0	0.248189	0.4146	0.0049	0.4146	0.1273	69	2%	Trusted
		1	0.245212	0.2467	0.9951	0.2467	0.0102	5812	98%	Distrusted
	TrustMe	0	0.507781	0.6205	0.0039	0.6205	0.0962	51	1%	Trusted
		1	0.496734	0.4975	0.9961	0.4975	0.0111	5830	99%	Distrusted

6.4.3. Silhouette Coefficient

Since external clustering validation methods are not applicable in our situation due to the absence of ground truth, the Silhouette coefficient as an internal validation method can be applied to measure the high similarity of an object to its cluster (cohesion) and the low similarity to other clusters (separation). The silhouette coefficient falls between -1 and +1. The maximum coefficient signifies that the object highly belongs to its cluster and weakly belongs to its contiguous clusters, whereas the minimum coefficient signifies that the object poorly belongs to its cluster [94]. The first following equation computes the silhouette coefficient *si* for each object xi, and the second equation computes the Silhouette coefficient (SC) as the mean values of *si* across all objects.

Table 7. Statistics for Five Clusters of the Three Trust Models.

Data	Model	Cluster	Initial Centroids	Final Centroids	Prior Probability	Normal Dist.		# of Instances	Percentage	Cluster Label
						Mean	*Std. Dev.*			
Bitcoin-Alpha	Raw Mean	0	0.159949	0.9458	0.0161	0.9458	0.0616	59	2%	Highly Trusted
		1	0.140666	0.4908	0.0641	0.4908	0.0768	249	7%	Trusted
		2	0.078926	0.2665	0.1679	0.2665	0.0463	628	17%	Barley Trusted
		3	0.021417	0.138	0.273	0.138	0.0344	1067	28%	Highly Distrusted
		4	0.001454	0.0258	0.4789	0.0258	0.0239	1780	47%	Distrusted
	PeerTrust	0	0.38983	0.44	0.033	0.44	0.0213	127	3%	Barely Trusted
		1	0.385604	0.1753	0.0029	0.1753	0.0705	30	1%	Highly Distrusted
		2	0.391762	0.8129	0.0018	0.8129	0.1268	6	0.002%	Highly Trusted
		3	0.387078	0.3882	0.953	0.3882	0.0078	3585	95%	Distrusted
		4	0.390971	0.5508	0.0092	0.5508	0.0412	35	1%	Trusted
	TrustMe	0	0.649086	0.7157	0.0195	0.7157	0.0259	83	2%	Trusted
		1	0.639352	0.2199	0.0021	0.2199	0.1561	30	1%	Highly Distrusted
		2	0.653882	0.8609	0.0058	0.8609	0.064	21	1%	Highly Trusted
		3	0.639882	0.6398	0.8028	0.6398	0.0093	3415	90%	Distrusted
		4	0.645654	0.6562	0.1697	0.6562	0.0083	234	6%	Barely Trusted
Bitcoin-OTC	Raw Mean	0	0.159678	0.2212	0.2117	0.2212	0.0511	1226	21%	Distrusted
		1	0.805003	0.8091	0.0875	0.8091	0.0808	486	8%	Highly Trusted
		2	0.001872	0.0512	0.2305	0.0512	0.0454	1337	23%	Highly Distrusted
		3	0.480177	0.3947	0.2559	0.3947	0.0537	1543	26%	Barely Trusted
		4	0.622767	0.5961	0.2144	0.5961	0.058	1289	22%	Trusted
	PeerTrust	0	0.248189	0.2872	0.0265	0.2872	0.0202	202	3%	Barely Trusted
		1	0.245212	0.1706	0.0059	0.1706	0.0454	84	1%	Highly Distrusted
		2	0.254436	0.4234	0.0031	0.4234	0.0571	18	0%	Trusted
		3	0.26546	1	0.0003	1	0.0177	1	0%	Highly Trusted
		4	0.245422	0.2462	0.9642	0.2462	0.0044	5576	95%	Distrusted
	TrustMe	0	0.507781	0.5708	0.0058	0.5708	0.0279	43	1%	Trusted
		1	0.496734	0.0572	0.0005	0.0572	0.0572	2	0%	Highly Distrusted
		2	0.499748	0.5113	0.0644	0.5113	0.0072	142	2%	Barely Trusted
		3	0.510235	0.8242	0.0007	0.8242	0.126	26	0%	Highly Trusted
		4	0.496743	0.4966	0.9286	0.4966	0.006	5668	96%	Distrusted

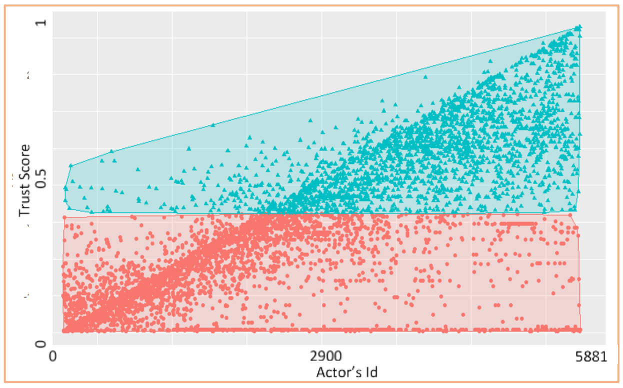
Table 8. Clustering Summary of All Models Using the Two Datasets.

Data	# of Clusters	Model	Log-Likelihood	Sum-Squared-Errors	# of Iterations
Bitcoin-Alpha	Bi-Level of Trust	Raw Mean	0.62508	41.7422	18
		PeerTrust	2.84695	1.6117	12
		TrustME	2.55354	2.2665	11
	Multi-Level of Trust	Raw Mean	0.84549	5.2760	34
		PeerTrust	3.29707	0.4784	45
		TrustME	3.06977	0.6106	24
Bitcoin-OTC	Bi-Level of Trust	Raw Mean	0.02252	101.8208	10
		PeerTrust	3.3507	1.0583	15
		TrustME	3.38126	0.9262	14
	Multi-Level of Trust	Raw Mean	0.0643	17.9733	14
		PeerTrust	3.8891	0.2974	32
		TrustME	3.74961	0.2980	22

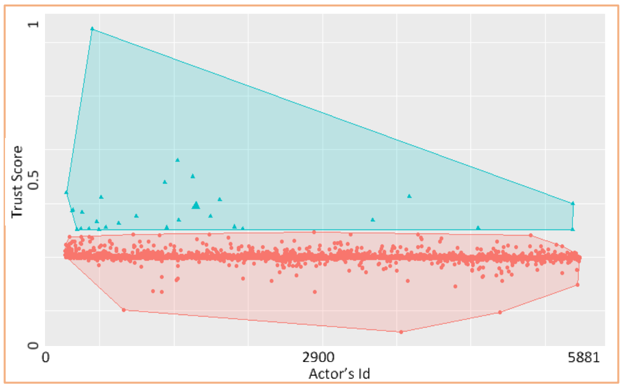
$$s_i = \frac{\mu_{out}^{min}(x_i) - \mu_{in}(x_i)}{max\{\mu_{out}^{min}(x_i), \mu_{in}(x_i)\}}$$

$$SC = \frac{1}{n}\sum_{i=1}^{n} s_i$$

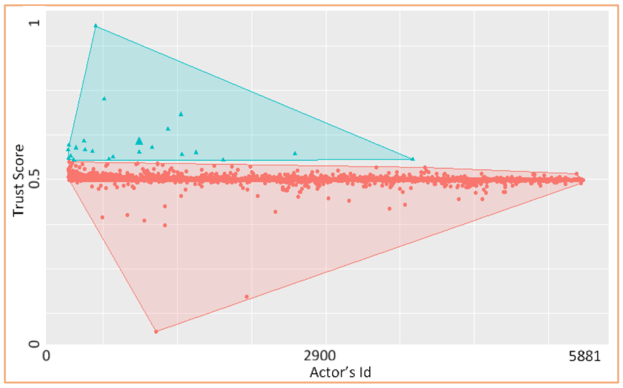
where $\mu_{in}(x_i)$ is the mean distance from x_i to points in its cluster; $\mu_{out}^{in}(x_i)$ is the mean distance from x_i to pints in its closest cluster.



a: RawMean Model using Bitcoin-OTC Dataset



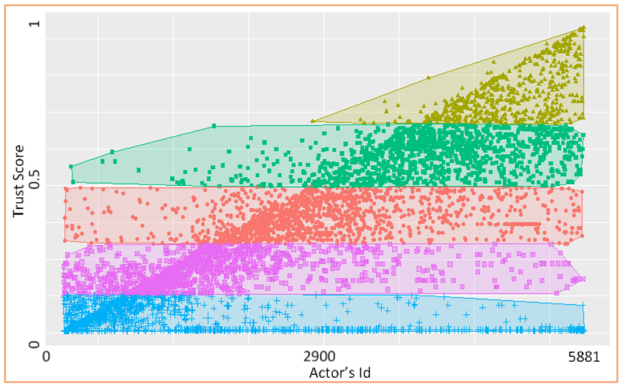
b: PeerTrust Model using Bitcoin-OTC Dataset



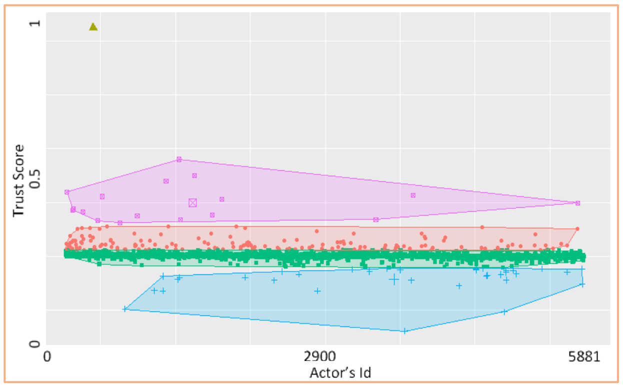
c: TrustMe model using Bitcoin-OTC dataset

Figure 26. Density-Base Clustering: Clusters of Trust Scores (Two Clusters)

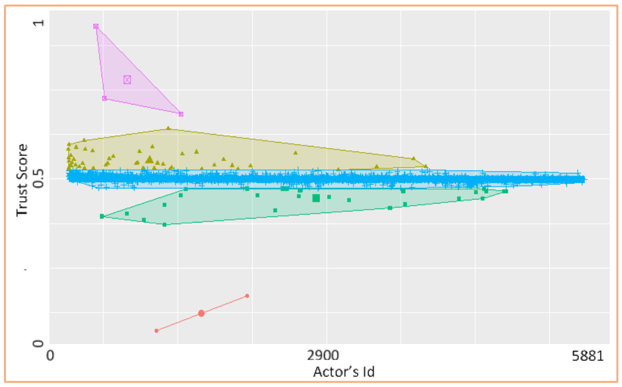
By computing silhouette coefficients for the thee model using the OTC-bitcoin dataset generating two clusters and then five clusters as shown in figure 28, we found that when $k=2$, the silhouette coefficient of RawMean was the worst at 0.60. While silhouette coefficients of PeerTrust and TrustMe were almost the same at 0.96. This means trust scores of TrustMe and PeerTrust are well-clustered as the overall silhouette close to 1. In the instance of $k=5$, the silhouette coefficient is decreased to be 0.56 which means the trust scores are poorly clustered. This also could imply that some trust scores are misplaced and assigned to closest clusters, or there is an overlapping clustering due to miscalculations. However, the overall silhouette of TrustMe outperforms other models with a value of 0.93 which is still close to 1. The silhouette coefficient of PeerTrust is also still high at 0.89. The large value indicates that there is a high-quality structure of the clusters as most trust scores sound to be assigned properly to its clusters. The quality of the generated clusters. Table 9 summarizes the Silhouette coefficient of all models using two different clustering settings (k=2 & 5).



RawMean Model using Bitcoin-OTC Dataset

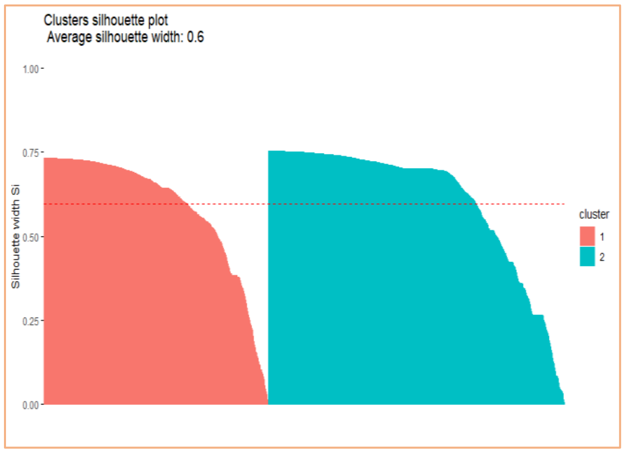


PeerTrust Model using Bitcoin-OTC Dataset

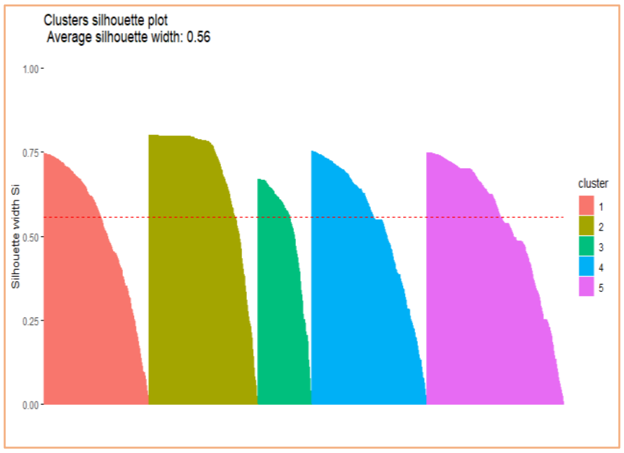


TrustMe Model using Bitcoin-OTC Dataset

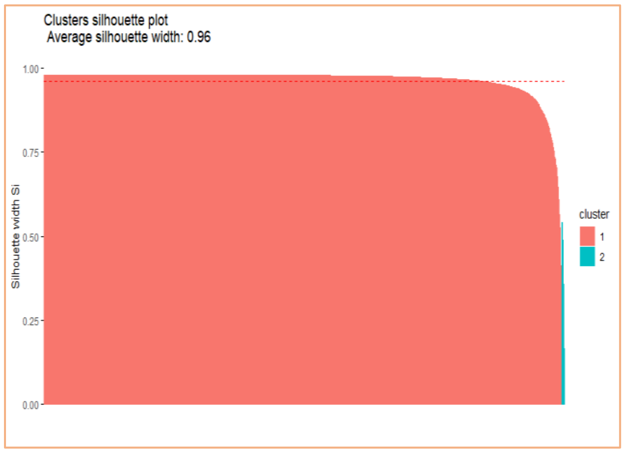
Figure 27. Density-Base Clustering: Clusters of Trust Scores (Five Clusters).



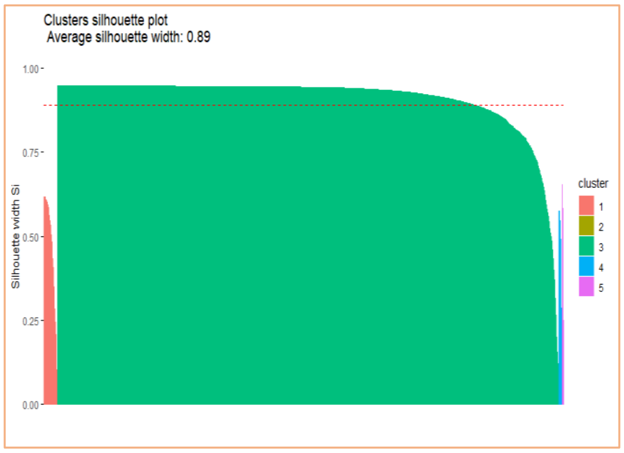
a. Cluster Silhouette of RawMean (2-clusters)



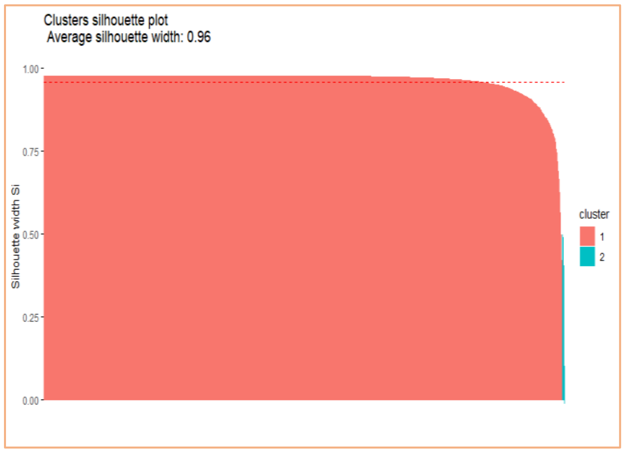
b. Cluster Silhouette of RawMean (5-clusters)



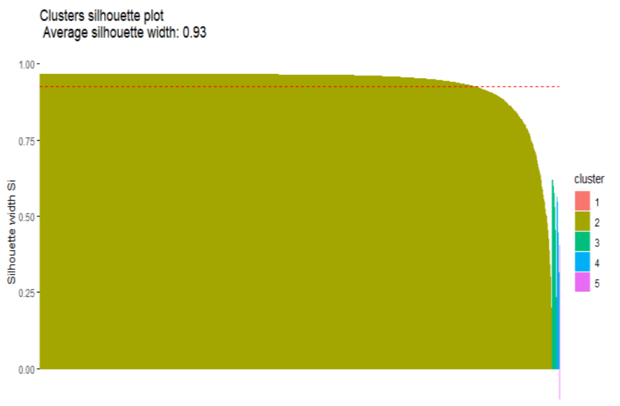
c. Cluster Silhouette of PeerTrust (2-clusters)



d. Cluster Silhouette of PeerTrust (5-clusters)



e. Cluster Silhouette of TrustMe (2-clusters)



f. Cluster Silhouette of TrustMe (5-clusters)

Figure 28. Cluster Silhouette of all Models using Two and Five Clusters.

Table 9. The Silhouette Coefficient of all Models using 2 Settings.

Models	Silhouette Coefficient	
	2-Clusters	5-Clusters
RawMean	0.5963307 ≡ 0.60	0.5567277 ≡ 0.56
PeerTrust	0.9599236 ≡ 0.96	0.8902622 ≡ 0.89
TrustMe	0.9584302 ≡ 0.96	0.9261229 ≡ 0.93

6.4.4. Correlation Analysis

We check for significant correlations between the trust factor as independent variables and trust score as a dependent variable, using the Spearman ρ coefficient. Spearman's correlation is a rank-based correlation measure that does not rest upon an assumption of normality, and it is robust concerning outliers. This analysis was conducted using bitcoin-Alpha dataset.

Table 10. Summary of Trust Factors against One Another.

First Side *vs.* Second Side	
Positive received median	Negative received median
Positive received median	Total number of positive received ratings
Positive received median	Total number of negative received ratings
Positive received median	Popularity factor
Negative received median	Total number of positive received ratings
Negative received median	Total number of negative received ratings
Negative received median	Popularity factor
Total number of positive received ratings	Total number of negative received ratings
Total number of positive received ratings	Popularity factor
Total number of negative received ratings	Popularity factor

6.4.4.1. Spearman's Correlations

In our trust model, we have the following independent factors (independent variables) that impact the trust score: positive rates median, negative rates median, number of positive ratings, number of negative ratings, and popularity. By using Spearman's correlation, we can see how each factor is correlated to each other, and how they are correlated to the generated trust scores.

- ***The Correlation of Independent Factors (Independent Variables)***

The purpose of this section is to study the correlation between the factors themselves as independent variables and how much each one correlated to the others. Table 10 summarizes the investigated factors against each other.

Table 11 and figures 29-38 demonstrate the correlations among the trust factors. Positive received median and negative received median are negatively correlated at -0.49 ($\rho = -0.49$) which means that when the positive received median increases, the negative received median decrease at $\rho = -0.49$ and vice versa. Another significant correlation exists between popularity and positive received median that are positively correlated at 0.83 ($\rho = 0.83$), so when the positive received median grows, the value of popularity factor increases as well. In contrast to the positive received median, the negative received median is negatively correlated to the popularity factor at -0.58 ($\rho = -0.58$). It is also evident that the popularity factor is positively correlated to the number of negative received ratings but at a very marginal value of 0.098 ($\rho = 0.098$).

Table 11. The Correlation among Independent Variables.

	Pos. Median	Neg. Median	# of PR	# of NR	Popularity
Pos. Median	-				
Neg. Median	-0.49	-			
# of PR	0.66	-0.62	-		
# of NR	0.09	-0.24	0.20	-	
Popularity	0.83	-0.58	0.89	0.098	-

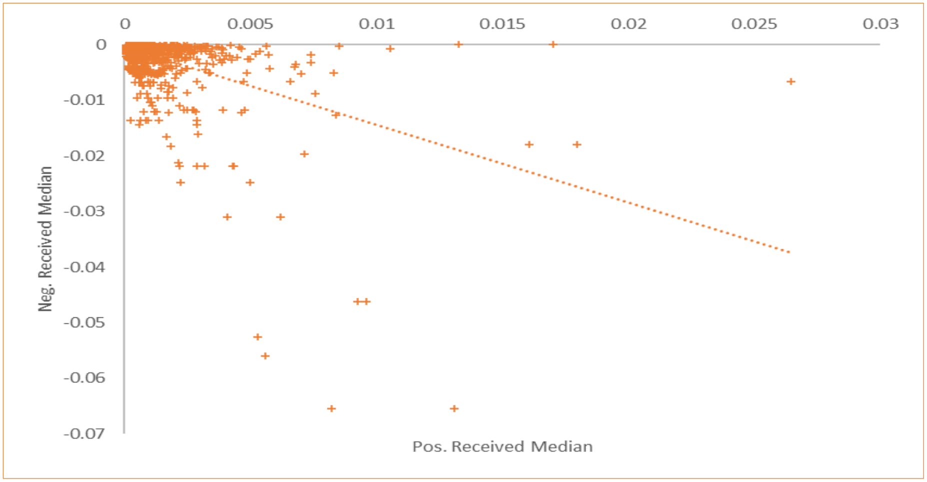


Figure 29. Correlation between Pos. Received Median *vs.* Neg. Received Median.

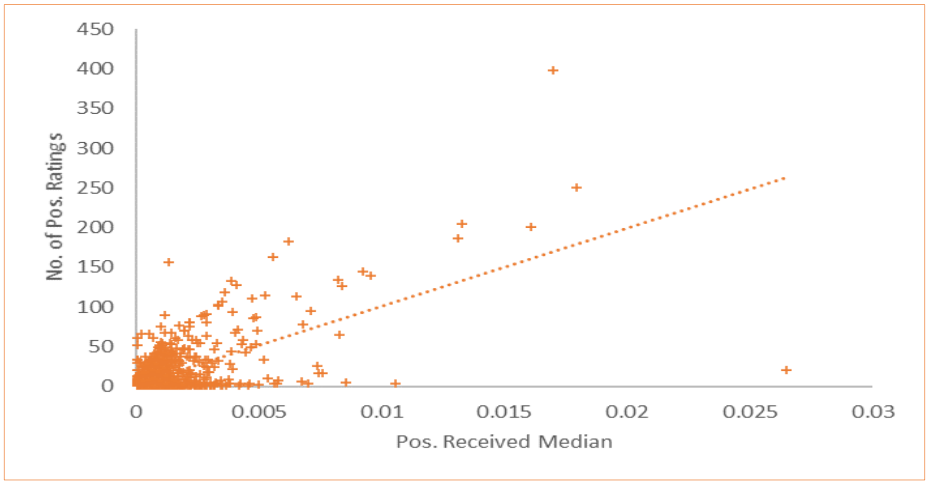


Figure 30. Correlation between Pos. Received Media *vs.* No. of Pos. Ratings.

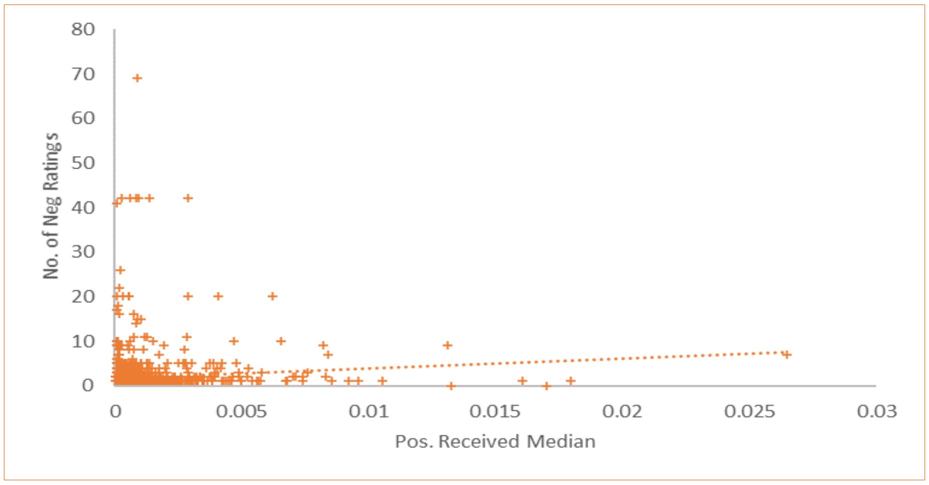


Figure 31. Correlation between Pos. Received Media *vs.* No. of Neg. Ratings.

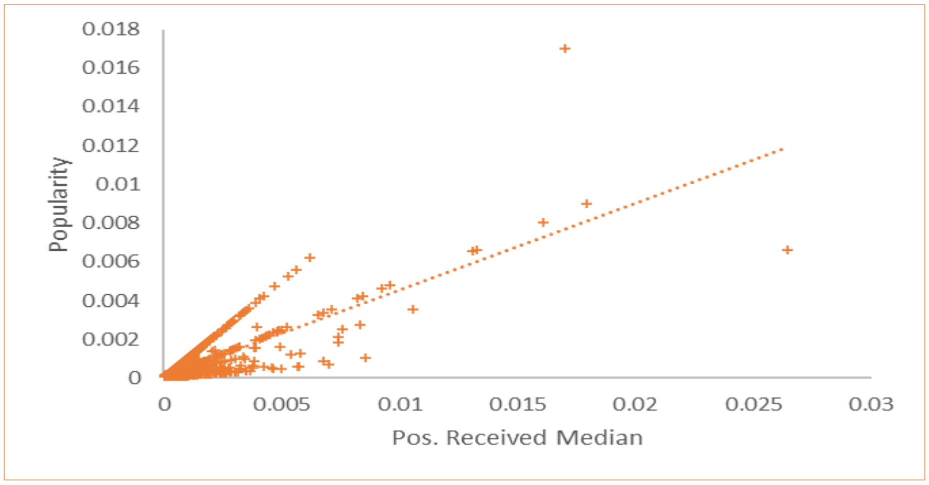


Figure 32. Correlation between Pos. Received Median *vs.* Popularity.

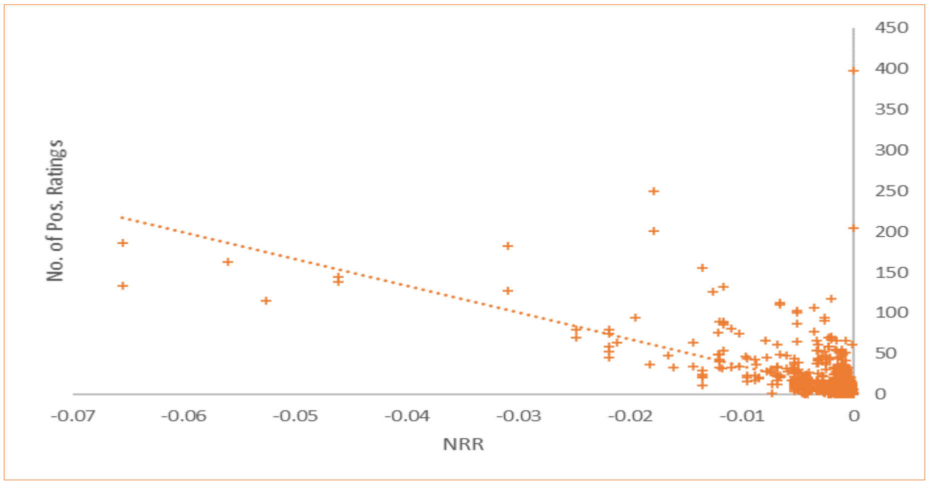


Figure 33. Correlation between Neg. Received Median *vs.* No. of Pos. Ratings.

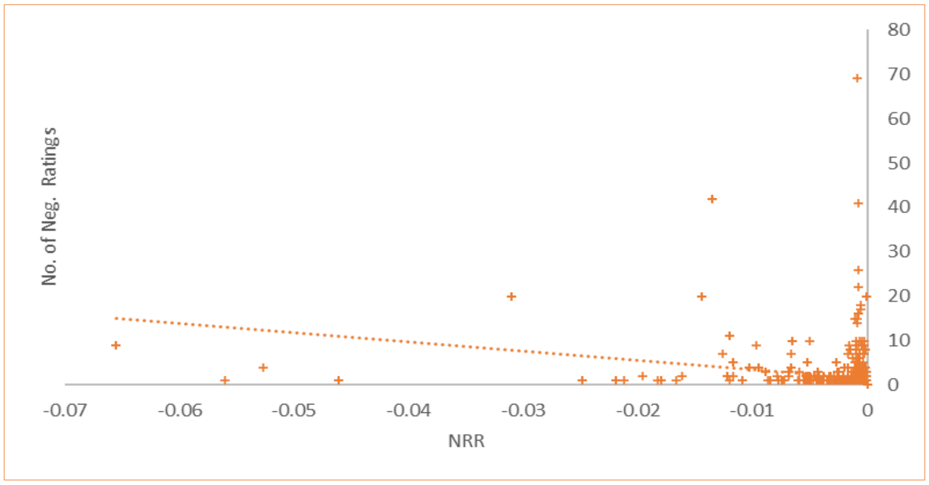


Figure 34. Correlation between Neg. Received Median *vs.* No. of Neg. Ratings.

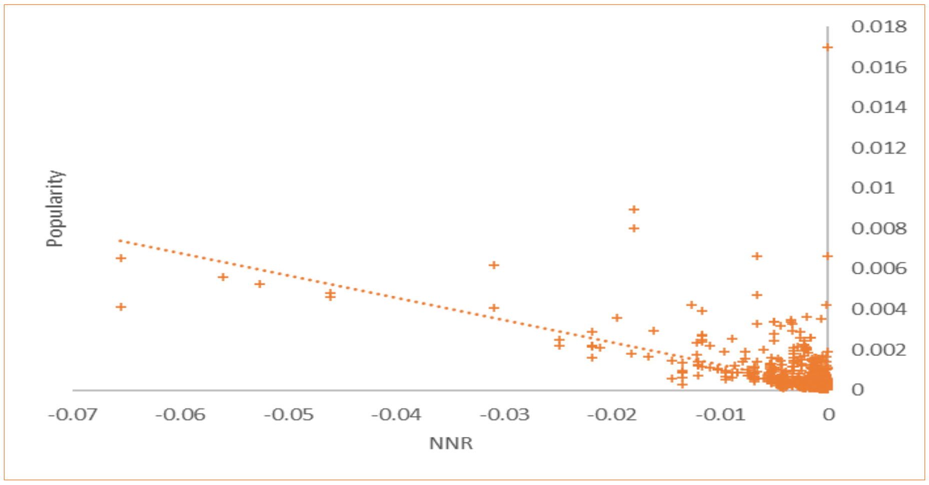


Figure 35. Correlation between Negative Received Median *vs.* Popularity.

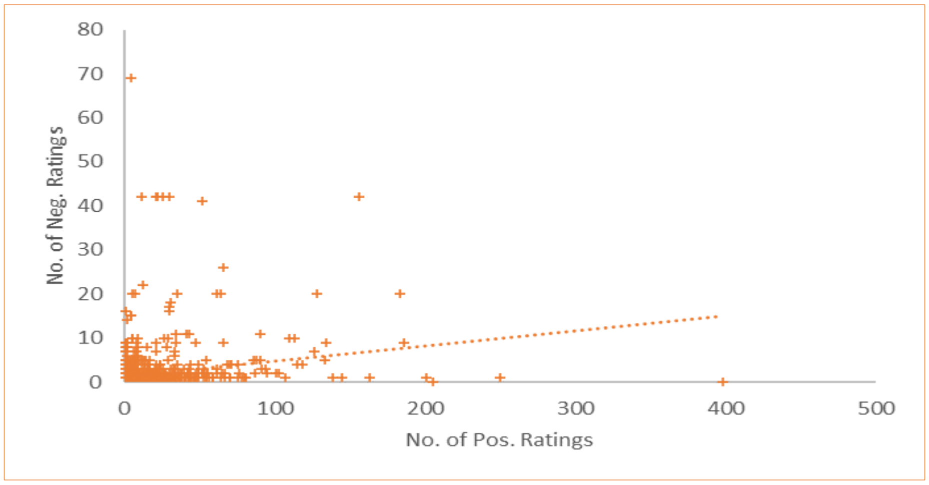


Figure 36. Correlation between No. of Neg *vs.* No. of Pos. Ratings.

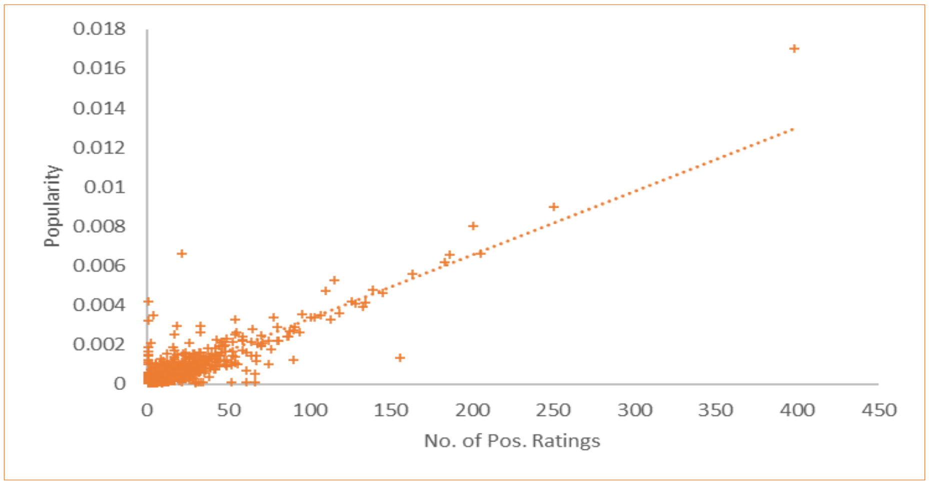


Figure 37. Correlation between No. of Pos. Ratings *vs.* Popularity.

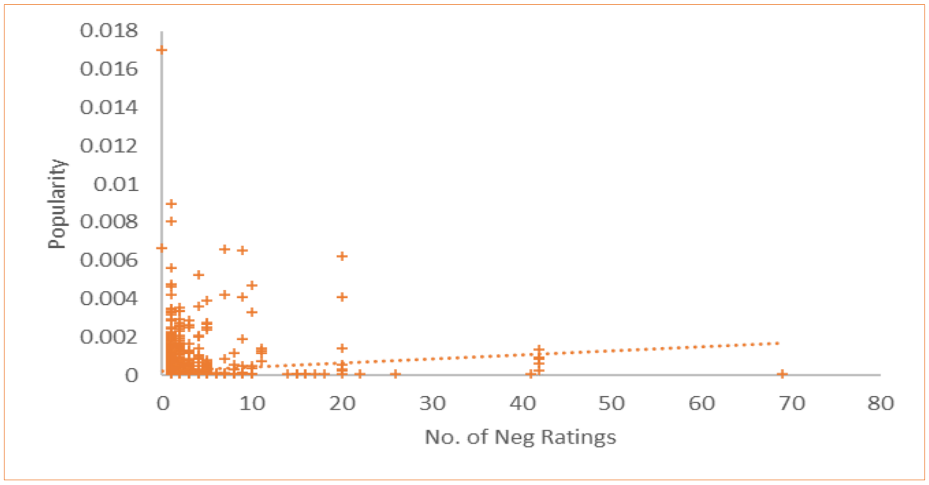


Figure 38. Correlation between No. of Neg. Ratings *vs.* Popularity.

- ***The Correlation of the Dependent variable and the Independent variables***

The outcome of the trust model is a trust score which is computed based on several independent factors. Here, we measure the spearman correlation between the trust score and all independent factors to observe the impact of each factor on the produced trust score.

Table 12. The Correlation between Independent Variables and Dependent Variable.

	Pos. Median	Neg. Median	# of PR	# of NR	Popularity
Trust Score	0.84	-0.27	0.64	-0.22	0.75

Table 12 and figures 39-43 illustrate the correlations between the trust score as a dependent variable and trust factors as independent variables. It is clear that the trust score is positively correlated with the positive received median at 0.84 ($\rho = 0.84$) which implies a strong correlation. Another strong correlation can be observed between the popularity factor and the dependent variable which is positively correlated at 0.75 ($\rho = 0.75$). However, there is a negative correlation between the dependent variable (trust score) and the negative received median at -0.27 ($\rho = 0.75$).

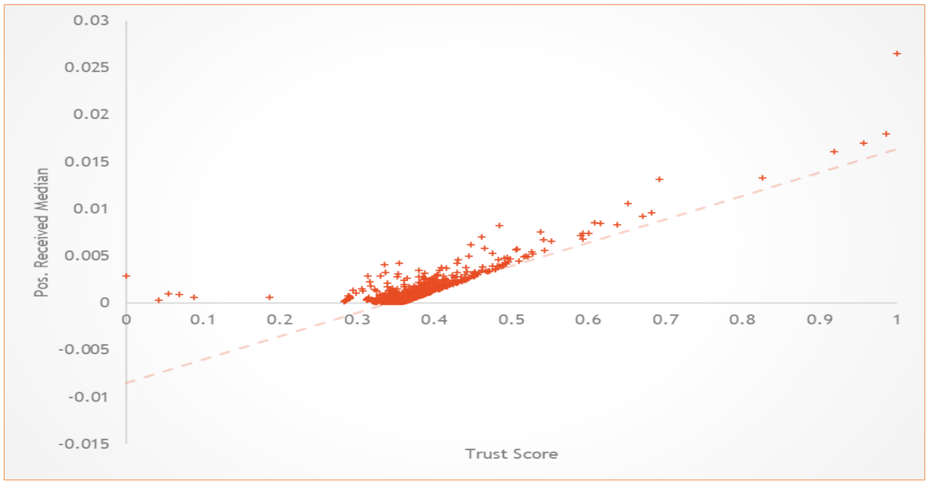


Figure 39. Correlation between Trust Score *vs.* Pos. Received Median.

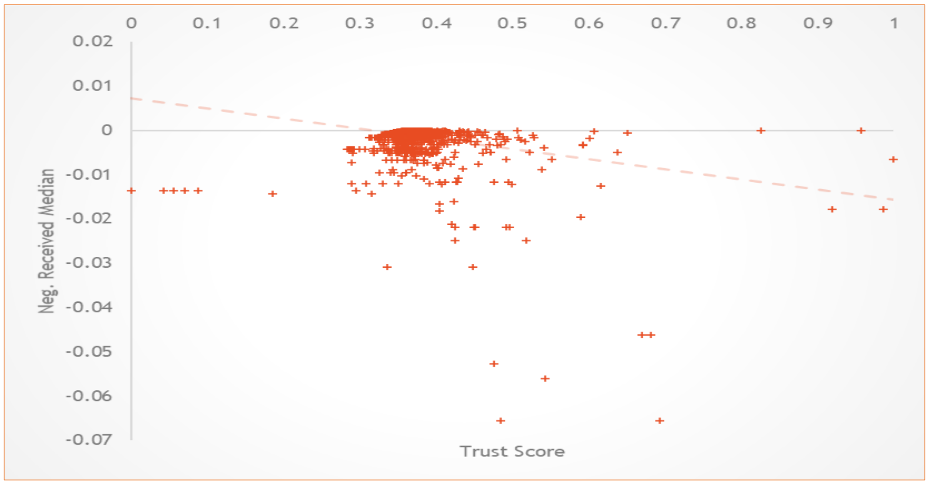


Figure 40. Correlation between Trust Score *vs.* Neg. Received Median.

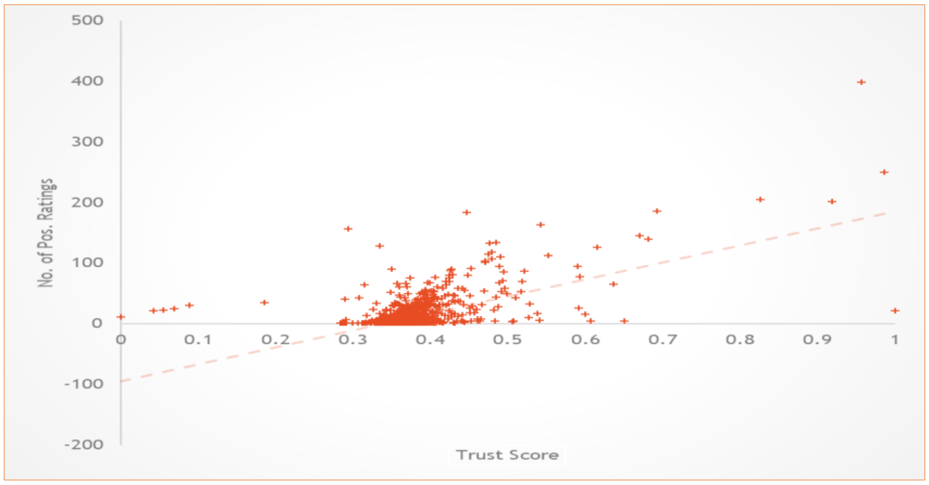


Figure 41. Correlation between Trust Score *vs.* No. of Pos. Ratings.

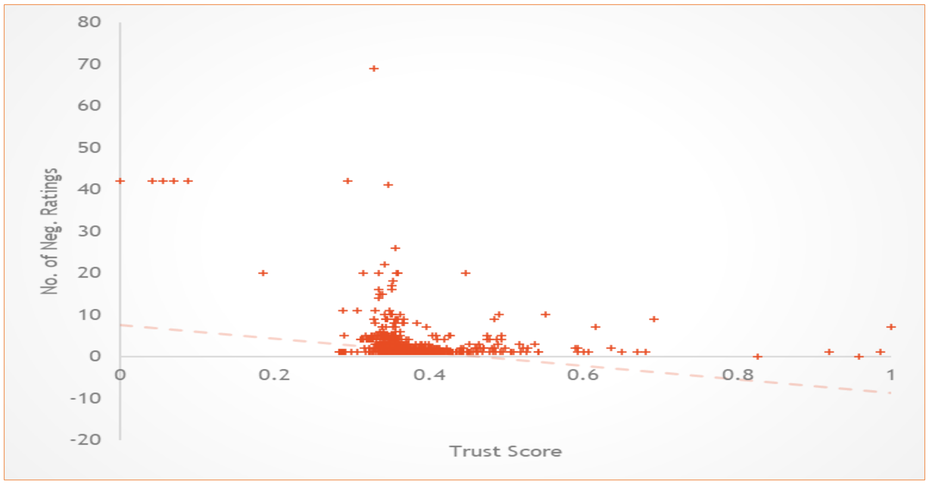


Figure 42. Correlation between Trust Score *vs.* No. of Neg. Ratings.

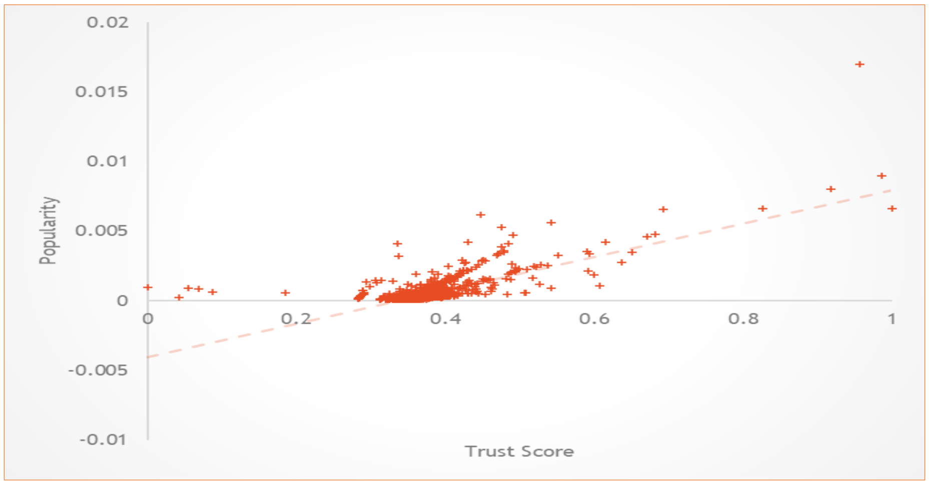


Figure 43. Correlation between Trust Score *vs.* Popularity.

This analytic-driven experiment operates as a pilot study to preliminary evaluate the trust models under study using unsupervised methodologies such as density-based clustering. The preliminary results exhibit that the RawMean model seems to be vulnerable by generating overlapped scores that are assigned mistakenly to unrelated groups, and there is a sign that our model (TrustMe) could be promising.

6.5. Graph Analysis - Community and Trust Analysis

In this study, the OTC-bitcoin dataset from [95, 96] was used as an example of an online community formed in a social network graphical model. The purpose is to disclose how actors interact with each other and how trust evolves through appending new interactions as edges over time. The interactions between actors can be represented in a directed graph $G = (V, E, W)$ where V denotes a finite nonempty set of vertices as actors, $E \subseteq V \times V$ indicates a set of direct associations (interactions) of ordered pairs of nodes/actors, and W denotes judgments in a form of weights. Some studies use a network edge to represent a direct trust, while in our work the edge only represents a direct judgment based on direct interaction between two parties. The phenomenon of trust evolvement remains vague and need to be comprehended, this draws the motivation of this section by providing some analysis as an attempt to understand the phenomenon. We, first need to investigate the topology of the given network as a large community G that may consist of many sub-communities/sub-graphs $G' = (V', E', W')$.

The generated trust score of the TrustMe model is mainly calculated based on the reputation score of an actor and some topological information about actors in a given social network. A topological perspective states that direct trust is occurred by a direct link between two actors, while indirect trust is occurred by indirect links. Direct trust is simple since it relies on one direct tie between two actors to generate a local trust score. The indirect trust is rather complex as it walks through many indirect ties between many actors to generate a global trust score as proposed in the TrustMe model.

Table 13. Network Topological Information.

Graph Metrics	Value
Graph Type	Directed
Vertices/Nodes/Actors/Users/Entities	5,881
Edges/Interactions	35,592
Diameter	11
Radius	1
Avg. Path length	3.79
Avg. Degree	6.052
Graph Density	0.001
Modularity	0.484
Avg. Number of Neighbors	7.309
Communities	22
Weakly Connected Components	4
Clustering Coefficient	0.149
Isolated Nodes	0
Self-loops	0
Multi-edge Node Pairs	4.005
Reciprocated Edge Ratio	0.56

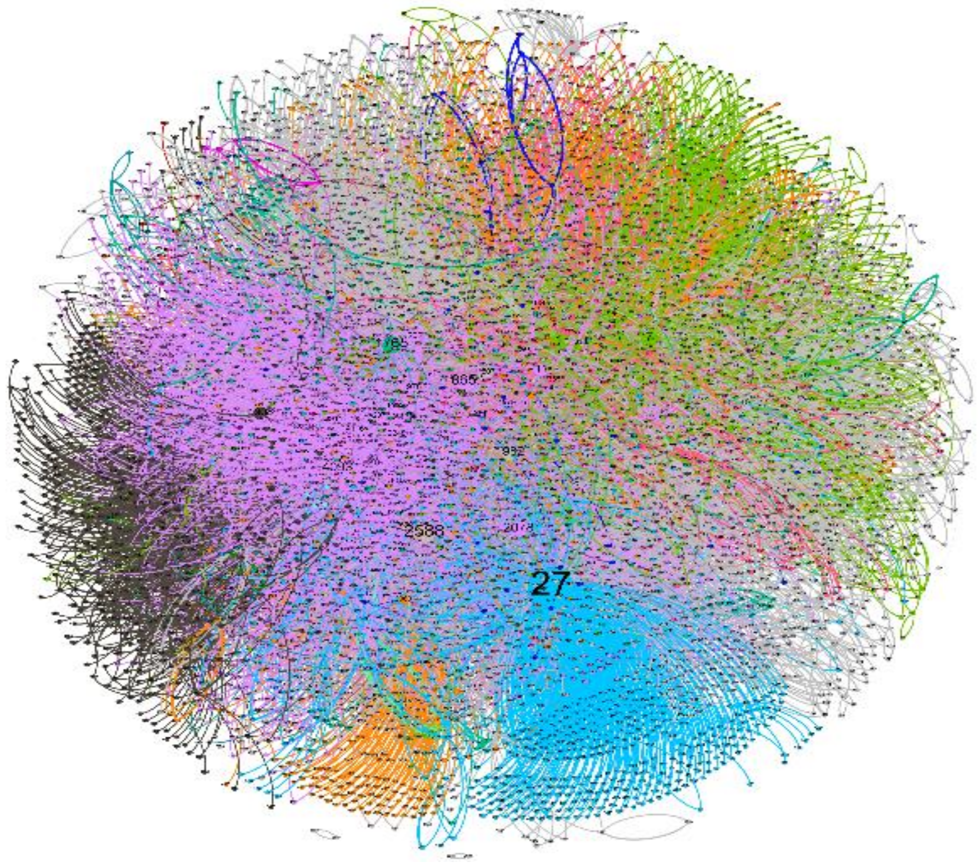


Figure 44. Network Graph.

6.5.1. Community Structure and Characteristics

Graph-theoretic concepts pave the road toward understanding and analyzing social phenomena such as trust and reputation [97, 98, 99]. The network in figure 44 is laid out using the "*Force-Atlas*" layout to cluster the nodes and then "*Fruchterman-Reingold*" to refine the layout to be circular. Betweenness centrality was employed to scale out the nodes' sizes. Table 10 shows that the network consists of 5,881 vertices, 35,592 edges, four weakly connected components. The diameter of the network is 11, and the average path length is 3.79. The relationships in this network are non-reflexive implies that no actor can evaluate oneself №(v_i, v_i), asymmetry alludes that $e(v_i, v_j) \neq e(v_j, v_i)$, and non-transitive implies that $\exists\, w(v_i, v_j) \wedge \exists\, w(v_j, v_k) \longrightarrow \exists\, w(v_i, v_k)$ is not necessarily true all the time.

6.5.2. Dynamics of Community Development

The term community at first glance indicates a geographical entity where there are precise boundaries such as a town, village, city, etc. However, a community can be a group of species that share some interests. Community dynamics is the process of transformation and evolvement by moving in or moving out a community. The term development sometimes postulates growth and expansion in a community. This is not always true since some people or entities may join or leave the community. The given network in this study shapes a community with actors/nodes that have shared interests such as exchanging bitcoins. New actors may join the community if they found that community conditions are appropriate for exchanging bitcoins or existing actors may leave the community if they found that the conditions are inappropriate. This can be connected to the rule of succession introduced by Pierre-Simon Laplace [100] that states if an action is repeated n times independently in succession, and get s successes and n-s failures, then the probability that it will occur again is $(s + 1)/(n + 2)$. This fundamental rule aids in driving the decision-making process which could be done intuitively or through reasoning and observation. For example, in a situation when an endless wave of fraud occurred in a community, it may be followed by a succession of actors leaving the community since it became unsafe and risky. To explore the structure of the given community and how it evolves and how actors affected by changes in the built environment, the network is divided into years from 2010 to 2016. Table 14 summarizes the statistics of the community dynamics, and figures 40 and 41 depict how the community metrics change.

Table 14. Summary of Community Statistics over Time.

G. Metrics	2010	2011	2012	2013	2014	2015	2016
Nodes/Actors	55	1637	3162	5161	5753	5879	5881
Edges/Interactions	142	7900	17332	30314	34539	35550	35592
Avg. Degree	2.582	4.826	5.481	5.874	6.004	6.047	6.052
Avg. Weighted Degree	7.109	8.396	7.925	6.028	6.006	6.113	6.125
Diameter	8	10	11	11	11	11	11
Avg. Path Len	3.24	3.87	3.798	3.75	3.73	3.719	3.718
Graph Density	0.048	0.003	0.002	0.001	0.001	0.001	0.001
Modularity	0.437	0.451	0.457	0.494	0.494	0.491	0.480
Possible Communities	6	11	13	15	15	17	22
Connected Components	2	2	2	3	3	4	4
Clustering Coefficient	0.066	0.099	0.120	0.136	0.144	0.149	0.149

6.5.2.1. Network Level Analysis

This section illustrates the information above focusing on the community growth as can be noticeable by comparing the community as a whole in figure 44 and the community in the first year (2010) shown in figure 47. It can be seen in figure 46 that from 2010 to 2011, the number of newly joined actors was 1,582 which represents around 2976% growth rate. The growth in 2012 was about twice the population of 2011 at 48%. The number of new actors joined the community in 2013 was 1999 which accounts for about 39% of the population. However, the numbers became progressively smaller from 2014 with 592 new members and 126 new members in 2015 representing 2% of the population, while in 2016, only two actors joined the community representing 0.034%of the population of 2016. Likewise actors, the number of interactions increases from 2010 through 2016 at the highest rate in 2013 with 12,982 new interactions, and at the lowest rate in 2016 with solely 42 new interactions.

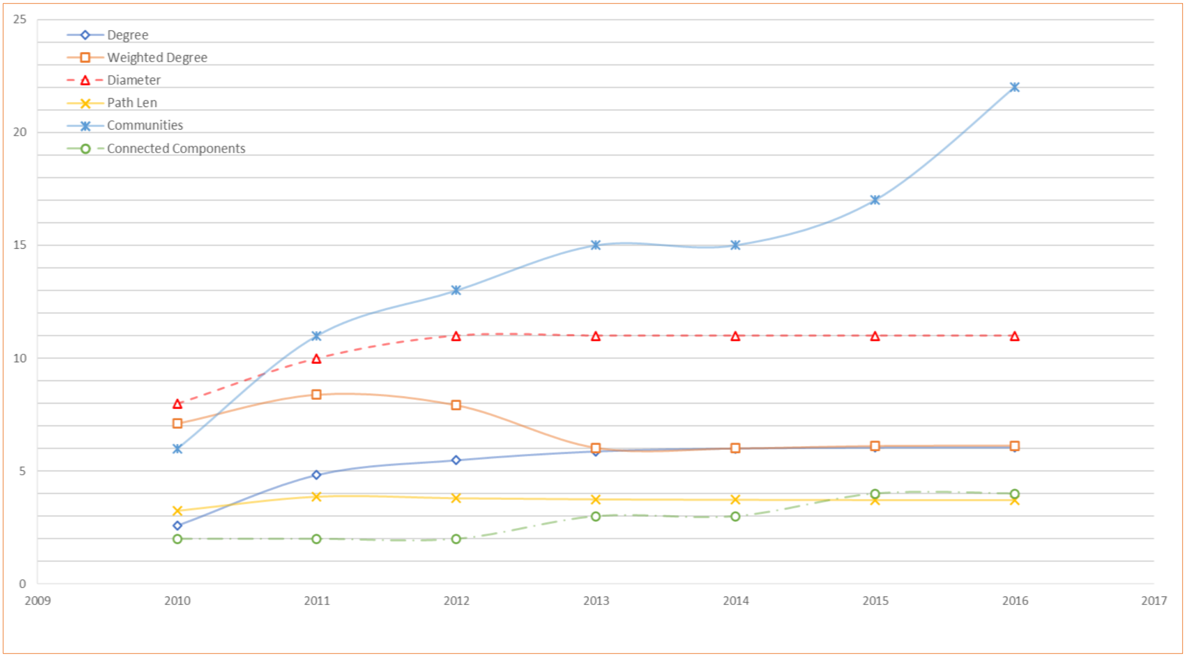


Figure 45. Evolvement of Network Topology.

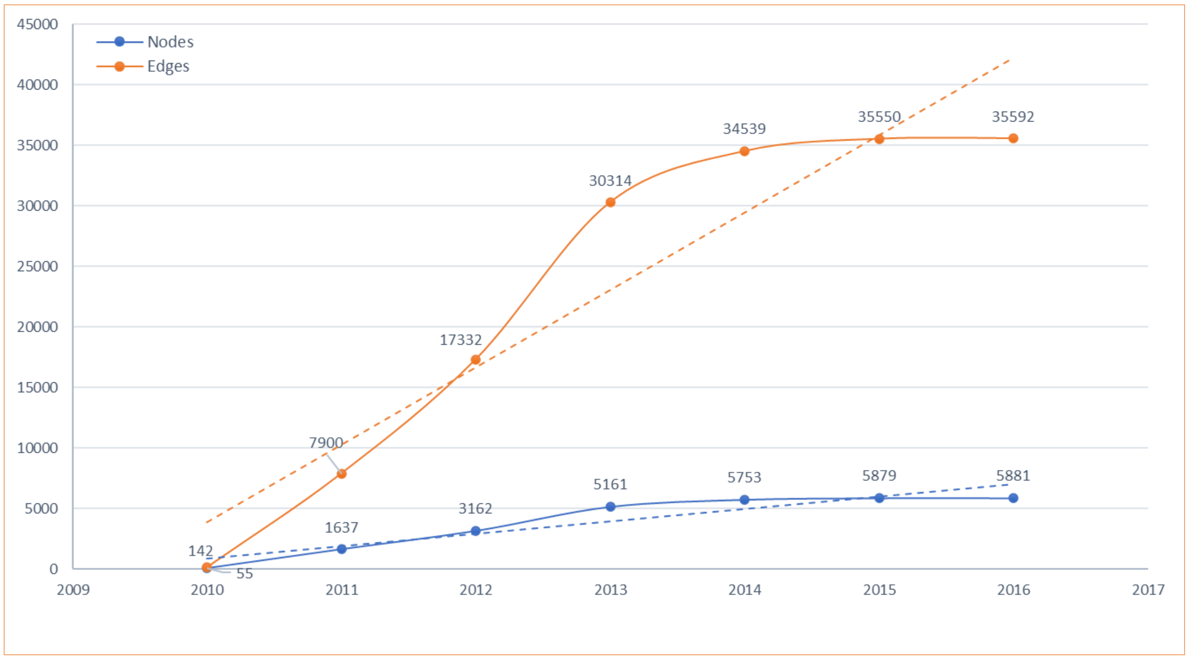


Figure 46. Evolvement of the Number of Actors and their Relationships.

This is an evident indication that the community becomes less attractive in the last two years as the rate of interaction is significantly dropped. Additionally, the value of the average degree changes slightly from 2014 to 2016. It is noticeable that the graph diameter as a macroscopic measure of the network width, which is the maximal distance between the pair of nodes, is apparently fixed at 11 from 2012 until 2016. This is another sign of low change in the network topology even after carrying out a large number of interactions in 2013 but merely representing an average of 0.019 of interactions for one actor throughout the year.

Figure 45 also depicts the degree centrality (interactions) that is quite small. Hence, the average of the weighted degree was declined in 2012 and thereafter became stable with a minor change. From table 14, it can be seen that the rate of relationship is at a low level in 2010 according to the average degree, and it had started to increase in the subsequent years. However, network metrics indicate that the relationship rate and the density score appear to be steady from 2013 until 2016 at 6 and 0.001 respectively. This demonstrates that solely a small proportion of possible interactions present in the network. This is perhaps actors tend to interact with other actors who they trust which usually takes time to develop particularly when the community increasingly becomes larger and complex.

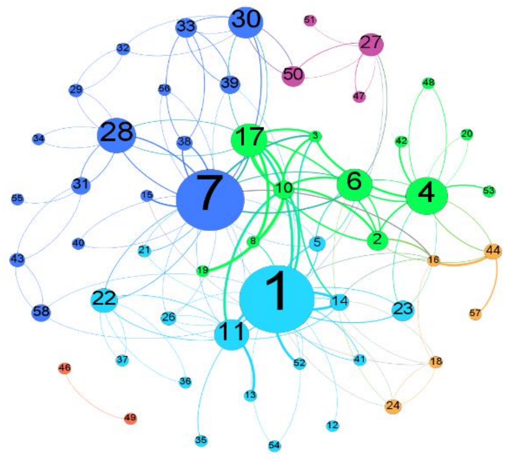


Figure 47. The Network/Community in 2010.

Table 15. The Network Characteristics of 2010.

Graph Metrics	Value
Graph Type	Directed
Vertices/Nodes/Actors/Users/Entities	55
Edges/Interactions	142
Diameter	8
Radius	1
Avg. Path length	3.24
Avg. Degree	2.582
Graph Density	0.048
Modularity	0.437
Avg. Number of Neighbors	3.309
Communities	6
Weakly Connected Components	2
Clustering Coefficient	0.066
Isolated Nodes	0
Self-loops	0
Multi-edge Node Pairs	51

6.5.2.2. Node Level Analysis

In this study, three types of degree centrality, and computed trust scores are employed to monitor and analyze the data of some actors. Thus, the top 10 actors were extracted for each metric involving actors receive a high score of positive in-degree, negative in-degree, betweenness centrality, eigenvector centrality, TrustMe score which is computed based upon the metric, and raw mean trust score which is a simple average. Degree centrality denotes the number of edges a node has to other nodes. The general concept of centrality is to grade nodes of a graph concerning how important each node is. Positive in-degree is the number of incoming edges that hold a positive weight (judgment scores), and negative in-degree is the number of incoming edges that hold a negative weight. Betweenness centrality is a general measure of the importance of a node in a network by quantifying how many times this node serves as a bridge between the shortest paths of many other nodes [101, 102, 103]. Eigenvector centrality is another approach that measures the importance of a node in a network by calibrating the importance of the neighboring nodes that linked to it. If the eigenvector-centrality value is high, it means that the node is connected to many other nodes that have high eigenvalues and so forth. The calculation of the eigenvector of a node is proportional to the sum of the eigen-centrality of the nodes that are linked to it [104].

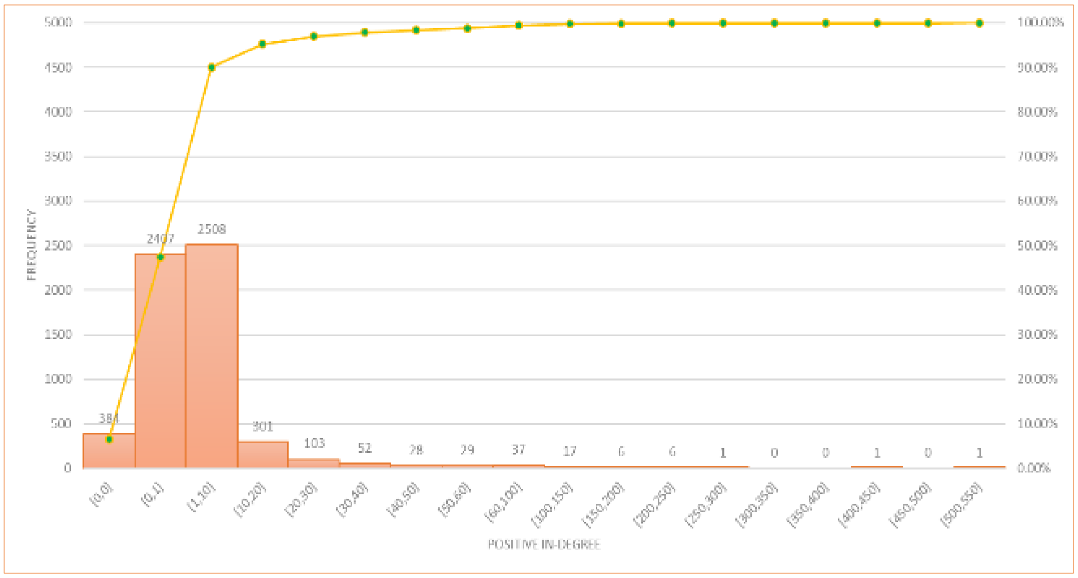


Figure 48. Positive In-Degree Distribution.

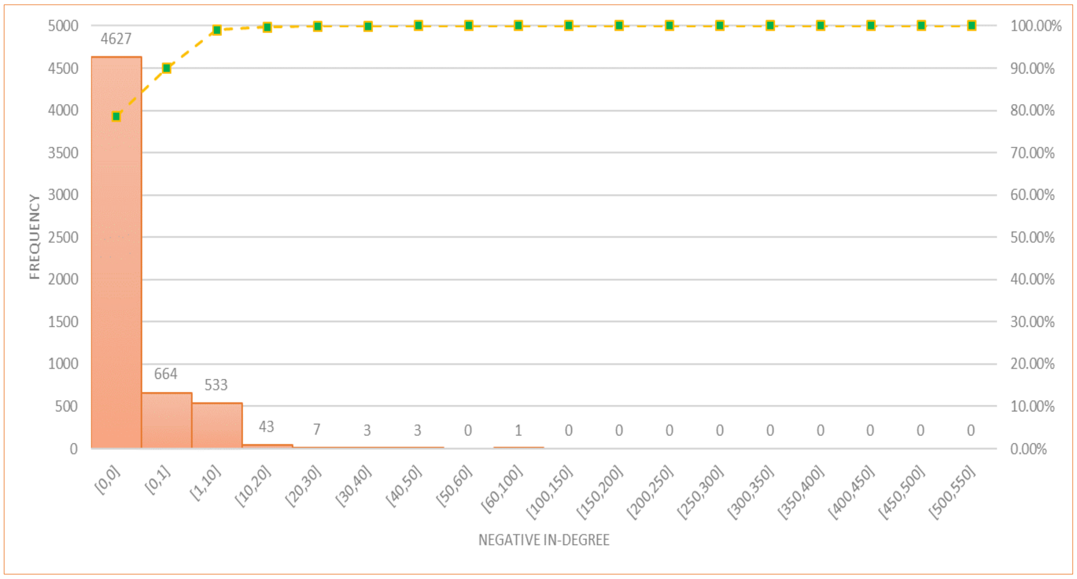


Figure 49. Negative In-Degree Distribution.

As shown in Figure 49, the main distribution of negative in-degree is right-skewed (positive skewness) concentrated around zero which is an indication of good behavior among most actors. In spite of that, there are several actors received negative evaluations. A population of 1,254 actors acquired negative feedback forming approximately more than 20% of the entire community. A number of 1,197 members received between 1 and 10 negative feedback, whereas 57 actors received negative feedback ranging from 10 to 100. This could be a sign of the prevalence of malicious behaviors in the community/network. Figure 48 exhibits the distribution of positive in-degree which is right-skewed (positive skewness) showing that 4915 members acquired a small number of positive evaluations ranging from 1 to10, which constitutes the majority. Additionally, 175 actors received a number of positive scores between 30-250, whilst merely three actors acquired a large number of positive evaluations falls between 250 and 550. By scrutinizing these statistics, relatively a small fraction of actors do not receive positive feedback. The large fraction, however, gained positive feedback but meager in quantity. Equally crucial, judgment score in terms of ratings ranging from -10 to 10 for each evaluation is not employed yet. The judgment score has a significant impact on the evaluation process. Namely, there is a necessity to monitor different elements and metrics to precisely appraise the community and discern well-behaved and malicious actors.

Table 16 summarizes the top 10 actors who obtain the highest score of the metrics mentioned above. To simplify reading betweenness centrality, it is normalized using a unity-based normalization function. The function rescales the original betweenness score into values ranging between [0,1] by considering the maximum value as the highest score of betweenness in the network 100% ≡ 1. Trust scores produced by the two metrics (TrustMe and RawMean) drive the analysis process. Table 13 (f) shows all the top 33 actors of the RawMean metric obtained 1.00 trust score which represents the ultimate trust. Through Conducting a quick examination, we found that all of those actors indeed receive solely one positive rating with an exception for actor# 4733 who receives two positive ratings. It is totally illogical when an actor can get an ultimate trust ≡ 1 through performing an orphaned interaction and receiving an orphaned positive rating. This analysis proves that the number of interactions does not affect the generated score of the RawMean metric. It is explicit that the RawMean metric, which remains used in many online communities, is highly vulnerable, and it can be exploited by dishonest actors to provide misleading information. Another indication is that no actor in the top 10 of the RawMean metric appears in the other top 10 tables such as betweenness and eigenvector centralities that evince the actors' importance in a given community. It demonstrates that no one in the top 10 of RawMean is important in the network depending on the rank of the top 10.

Due to the initial examination, the RawMean metric is eliminated from the analysis process. It is evident that this approach is not effective, and even without malicious acts, it may provide delusive information. However, TrustMe scores seem to be more akin and highly correlated to the scores of several network metrics. Actors who hold numbers (27, 2588, 1, 1765, 1982, and 4093) appears in TrustMe, in-degree$^+$, betweenness, and eigenvector measures. Whereas, actor# 7 appears in TrustMe, in-degree$^+$, betweenness tables and does not appear in eigenvector centrality. Actor# 4116 also, appears in TrustMe, in-degree$^+$, and eigenvector but not in betweenness centrality. Actor# 2078 presents in TrustMe and betweenness tables, while actor# 978 exhibits only in TrustMe table.

Actors who hold questionable scores such as actor# 1982 who appears in all tables need to be placed under investigation. The actor obtained 234 positive ratings that representing 84% of all received ratings, while acquired 45 negative ratings representing 16% of the entire received ratings. By adding the weights of these ratings, we found that actor# 1982 earned 544 positive weighted ratings and -342 negative weighted ratings. This discloses that the negative ratings, in fact, represent 39% of the entire received ratings and the positive ratings represent 61% of the entire received ratings. Although the TrustMe model has its own punishment mechanism, it might be overly tolerant. Therefore, there is an imperative call to inject a proper punishment engine into the trust model to reduce the trust score when the negative feedback forms a substantial portion of the gained evaluations yet after removing dishonest feedback. It is important to keep in mind that the TrustMe model employs a popularity factor that stands for PageRank. Popularity factor works as a weight that raises or downs the rating score. The interpretation of this situation may be due to that the positive ratings were weighed by actors with high popularity whereas the negative ratings were weighed by actors with a low popularity score.

Table 16. The Top 10 Trust Scores with Network Metrics.

a: Top 10 In-Degrees$^+$			b: Top 10 In-Degrees$^-$			c: Top 10 Betweenness		
Actor	in.degree$^+$	TrustMe	Actor	in.degree$^-$	TrustMe	Actor	Betweenness	TrustMe
27	535	0.89	3669	75	0.00	27	1.000	0.89
2588	411	0.74	1982	45	0.63	2588	0.438	0.74
1765	270	0.65	1342	45	0.47	1765	0.349	0.65
1982	234	0.63	1765	41	0.65	865	0.349	0.62
1	226	0.68	865	38	0.62	1	0.317	0.68
865	226	0.62	2447	36	0.24	4093	0.295	0.63
7	216	0.67	1971	33	0.33	2078	0.293	0.64
4093	211	0.63	3820	26	0.55	7	0.280	0.67
4116	203	0.66	792	26	0.52	1982	0.272	0.63
11	190	0.62	1999	25	0.55	1908	0.227	0.59

d: Top 10 Eigenvector			e: Top 10 TrustMe Scores			f: Top 10 RawMean Scores		
Actor	eigenvector	TrustMe	Actor	TrustMe	TrustMe	Actor	RawMean	TrustMe
2588	1.000	0.74	27	0.89	0.89	490	1.00	0.51
865	0.927	0.62	2588	0.74	0.74	774	1.00	0.51
27	0.885	0.89	1	0.68	0.68	1082	1.00	0.51
1765	0.830	0.65	7	0.67	0.67	1221	1.00	0.51
1982	0.738	0.63	4116	0.66	0.66	1286	1.00	0.51
4093	0.706	0.63	1765	0.65	0.65	1299	1.00	0.51
1	0.692	0.68	2078	0.64	0.64	1459	1.00	0.51
4207	0.643	0.60	978	0.64	0.64	1503	1.00	0.51
1294	0.635	0.60	1982	0.63	0.63	1618	1.00	0.51
4116	0.612	0.66	4093	0.63	0.63	24 more	1.00	0.51

Actors who have high betweenness scores are deemed vital nodes in the network that functions as bridges controlling the evaluation flow between communities. Since TrustMe metric supplies global trust score (indirect trust) relying on local evaluations (direct interaction), the indirect evaluation significantly flows through nodes with high betweenness. Through exploring the top 10 betweenness and TrustMe scores, it is found that approximately 28% of the population has betweenness centrality above the average, and eight of the actors who are ranked in the top 10 of TrustMe are also among the ranked top 10 of betweenness centralities. An exception is for actors 865 and 1908 who are ranked in the top 20 of TrustMe and do not appear in the top 10 list. In figure 3, the node size represents betweenness centrality which is noticeable because the community was small in 2010. It can conspicuously be seen that there is an isolated sub-community consists of only two actors 46 and 49 with a sole transaction/interaction. The two actors have zero betweenness score which implies that they are not highly important in the network, but further investigation is required to ensure that both actors are not malicious. Thus, actors with high betweenness are responsible for managing the evaluation information flow between the actor itself and other actors in the community. If those actors are malicious, they may pass misleading information between communities. Nevertheless, the TrustMe model shows that those actors earn the highest trust score. Hence, it can be concluded that this is an indicator of a healthy community.

Another crucial measure is eigenvector centrality. By scanning the top 10 eigenvector and TrustMe scores, it is discovered that approximately 24% of the population has eigenvector centrality above the average, and seven of the actors who are ranked in the top 10 of TrustMe are also ranked in the top 10 of eigenvector centralities. Actor 978 is ranked eleventh in eigenvector and not shown in the top 10. Another point is that a correlation between in-degree$^+$ and eigenvector can be caught as eigenvector depends on its neighbors and how well-connected a node is. Eight actors appeared in the two lists of in-degree$^+$ and eigenvector but the other missed two: actor 11 is ranked twelfth in eigenvector and actor 7 is ranked 22nd. The number of edges/ neighbors can help in revealing malicious behavior in a given community and precisely evaluate actors' trustworthiness. Sometimes eigenvector centrality is considered a trust measure as it relies on the quality of the relationships. Eigenvector alone, However, can be circumvented by misbehaved actors through establishing a large fraction of fake interactions.

6.5.2.3. Observing Trust Evolvement:

For trust evolvement traceability, the dataset is divided into years which generates seven sub-datasets from 2010 through 2016. The selection process for tracking specific nodes/actors was carried out by first picking an actor with the maximum value of each measure based on the information of the last year (2016) with ignoring RawMean. Likewise, an actor with the maximum number of negative evaluations plus two more actors from the negative in-degree table with the lowest and highest TrustMe score is chosen. Thus, the actors that are subject to more investigation are 27, 2588, 3669, 2447, and 1765. Table 17 summarizes all the chosen actors.

Table 17. Summary of the Chosen Actors.

Actor	Table	Description
actor# 27	{a, c, & e}	Maximum score
actor# 2588	{d}	Maximum score
actor# 3669	{b}	Maximum score
actor# 1765	{b}	Highest TrustMe score with high number of negative evaluations
actor# 2447	{b}	Second lowest TrustMe score with high number of negative evaluations

The following tables (table 18) track the change of the chosen actors over time represented in years (2010-2016). Each table recaps the calculated scores of in-degree+, in-degree-, betweenness, eigenvector, PageRank, RawMean, and TrustMe metrics in seven tables. For simplicity, PageRank scores were normalized using unity-based normalization function as it was done with betweenness centrality, so the maximum PageRank score is normalized as 1.00 and the minimum score is normalized as 0.00.

Table 18. Actors' Scores over Years.

2010 a: The chosen actors' scores in 2010

Actor#	W.in-Degree$^+$	W.in-Degree$^-$	Betweenness	Eigenvector	PageRank	Raw.Mean	TrustMe
27	4	0	0.20	0.02	0.13	0.60	0.55
1765							
2447				*Not Yet Exist*			
2588							
3669							

2011 b: The chosen actors' scores in 2011

Actor#	W.in-Degree$^+$	W.in-Degree$^-$	Betweenness	Eigenvector	PageRank	Raw.Mean	TrustMe
27	150	0	0.56	0.39	0.59	0.57	0.72
1765							
2447				*Not Yet Exist*			
2588							
3669							

2012 15.c: The chosen actors' scores in 2012

Actor#	W.in-Degree$^+$	W.in-Degree$^-$	Betweenness	Eigenvector	PageRank	Raw.Mean	TrustMe
27	448	0	1.00	0.82	1.00	0.58	0.84
1765	277	-30	0.37	0.51	0.50	0.58	0.64
2447	0	-118	0.00	0.07	0.04	0.11	0.00
2588	133	0	0.15	0.23	0.25	0.59	0.59
3669				*Not Yet Exist*			

2013 d: The chosen actors' scores in 2013

Actor#	W.in-Degree$^+$	W.in-Degree$^-$	Betweenness	Eigenvector	PageRank	Raw.Mean	TrustMe
27	830	0	1.00	0.84	1.00	0.59	0.88
1765	502	-371	0.35	0.70	0.47	0.53	0.63
2447	39	-191	0.00	0.16	0.04	0.26	0.31
2588	1011	-2	0.53	1.00	0.81	0.63	0.75
3669	50	-665	0.03	0.29	0.10	0.09	0.00

2014 e: The chosen actors' scores in 2014

Actor#	W.in-Degree$^+$	W.in-Degree$^-$	Betweenness	Eigenvector	PageRank	Raw.Mean	TrustMe
27	966	0	1.00	0.84	1.00	0.60	0.89
1765	539	-383	0.35	0.71	0.45	0.53	0.65
2447	39	-277	0.00	0.20	0.05	0.22	0.23
2588	1043	-2	0.48	1.00	0.75	0.63	0.75
3669	50	-725	0.03	0.32	0.10	0.08	0.00

Table 18. Actors' Scores over Years (continued).

2015 f: The chosen actors' scores in 2015

Actor#	W.in-Degree$^+$	W.in-Degree$^-$	Betweenness	Eigenvector	PageRank	Raw.Mean	TrustMe
27	1016	0	1.00	0.89	1.00	0.60	0.89
1765	604	-385	0.35	0.82	0.46	0.54	0.65
2447	40	-296	0.00	0.25	0.05	0.22	0.24
2588	1043	-2	0.44	1.00	0.72	0.63	0.74
3669	50	-725	0.02	0.33	0.09	0.08	0.00

2016 g: The chosen actors' scores in 2016

Actor#	W.in-Degree$^+$	W.in-Degree$^-$	Betweenness	Eigenvector	PageRank	Raw.Mean	TrustMe
27	1016	0	1.00	0.89	1.00	0.60	0.89
1765	615	-385	0.35	0.83	0.46	0.54	0.65
2447	40	-296	0.00	0.25	0.05	0.22	0.24
2588	1043	-2	0.44	1.00	0.72	0.63	0.74
3669	50	-725	0.02	0.33	0.09	0.08	0.00

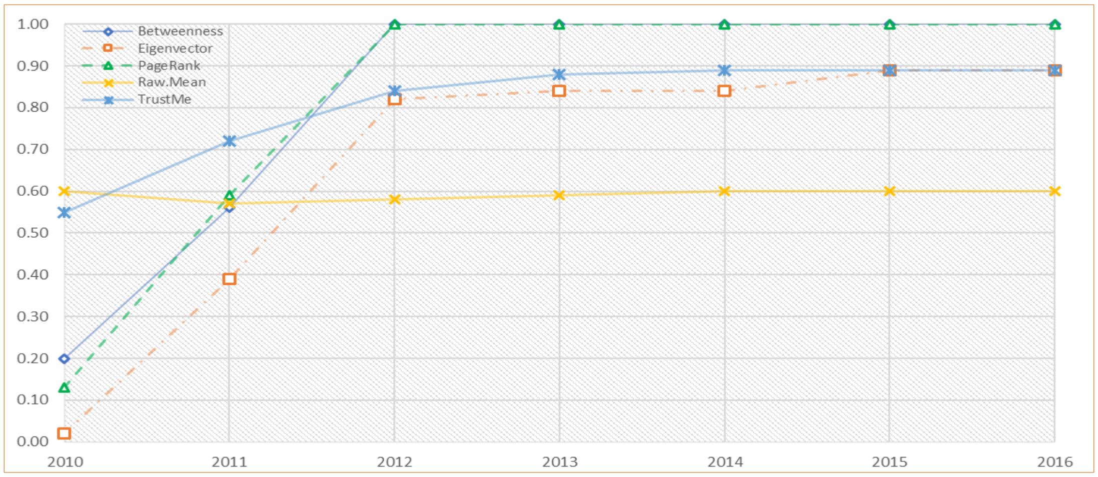


Figure 50. Evolvement of Actor's Scores over Years: Actor# 27.

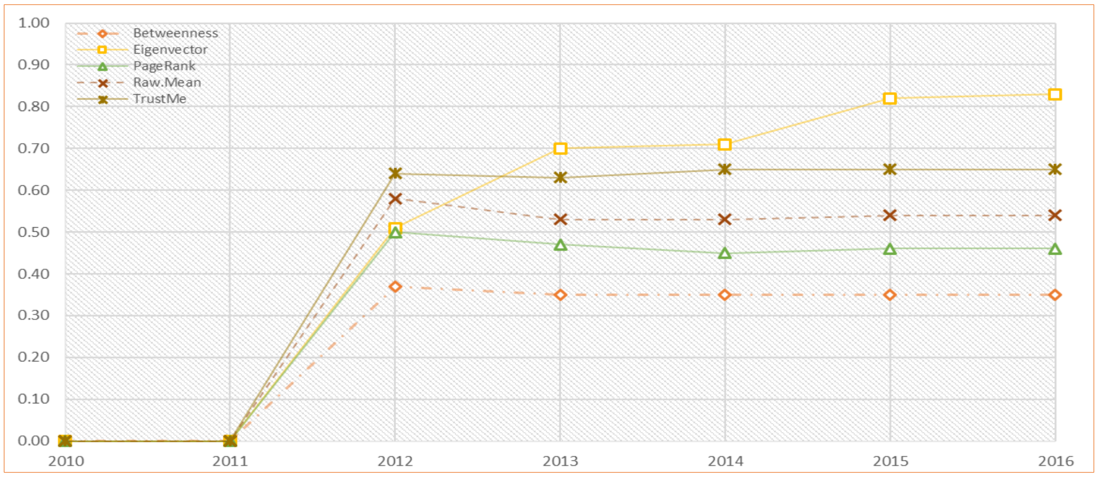


Figure 51. Evolvement of Actor's Scores over Years: Actor# 1765.

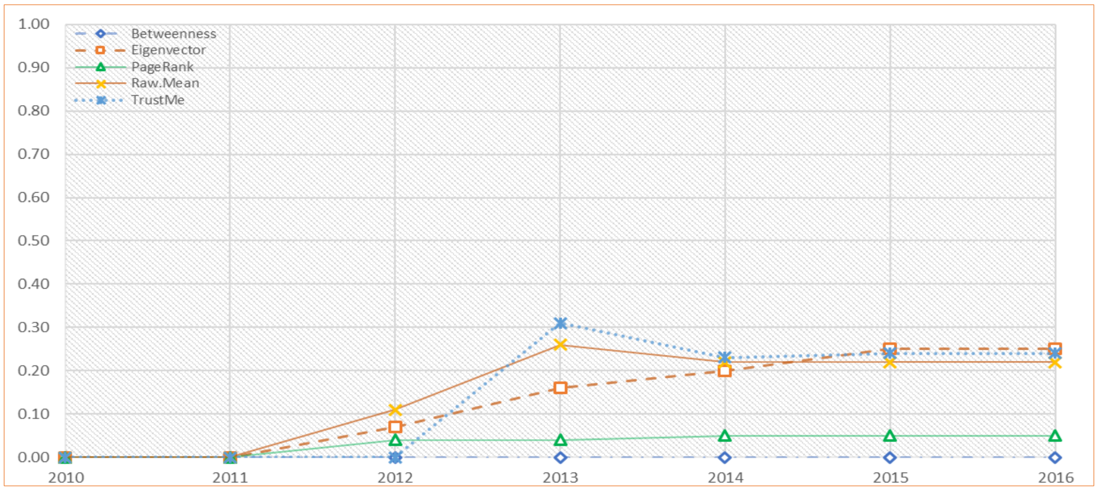


Figure 52. Evolvement of Actor's Scores over Years: Actor# 2447.

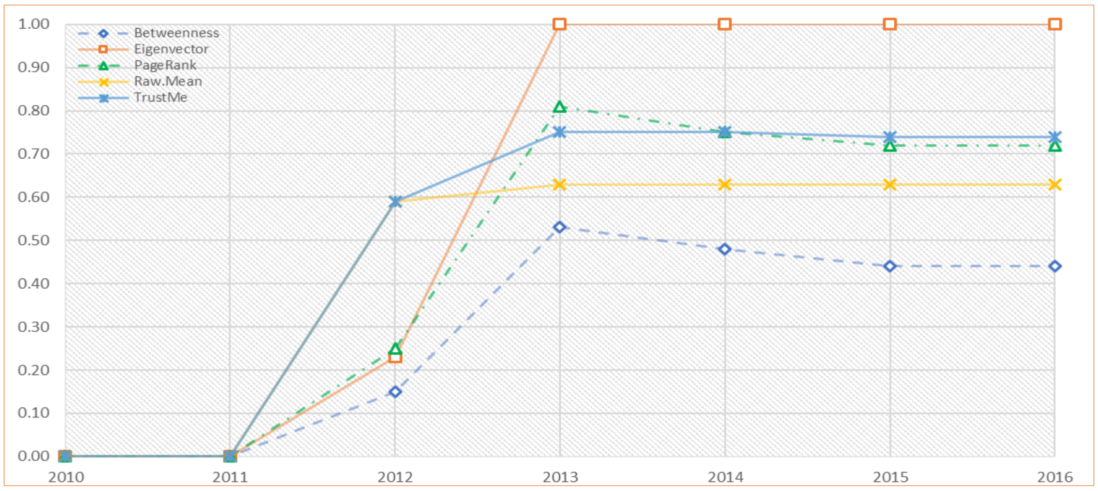


Figure 53. Evolvement of Actor's Scores over Years: Actor# 2588.

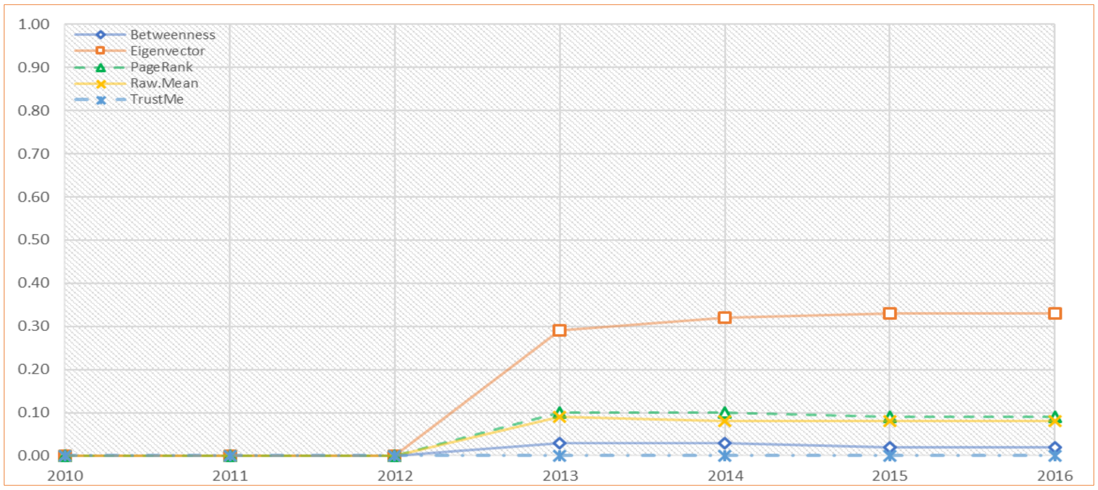


Figure 54. Evolvement of Actors' Scores over Years: Actor# 3669.

Figures 50-54 show the evolvement of trust along with topological information for every chosen actor. Actor 27 has the highest TrustMe, betweenness, and PageRank scores from 2012 until 2016. Actor 27 received a high eigenvector score, but it is not the largest score. The TrustMe score of this actor developed from 2010 until 2016 received scores of 0.55, 0.72, 0.84, 088, and 0.89 respectively and Ascendingly, while RawMean scores go up and down starting with 0.60 and ending with 0.60 with values around the fifties in between. It is noticeable in figures 1 and 3 that actor 27 is the most important one with the greatest betweenness score. Actor 2588, however, received the highest eigenvector score and the second highest TrustMe score. Even though the weighted positive in-degrees is the largest value owned by actor 2588, this actor received a single negative evaluation that reduces its trust score and ranked second. Compared with the RawMean metric, actors who generally obtained the highest score of TrustMe, eigenvector, PageRank, and betweenness such as actors 27 and 2588, received RawMean scores around the sixties. For instance, actor 27 received 535 positive evaluations with zero negative evaluation during the seven years, RawMean gave this actor 0.60, while it gave actors with only one interaction a full trust score (1.00). TrustMe, in contrast, seems to be fairer and correlated with other metrics since they move toward the same direction particularly when an enormous amount of interactions take place. It sounds like actors 27 and 2588 the most trusted ones, while actors 3669 and 2447 are the most distrusted ones according to the given data. Actor 3669 received a low trust score either with TrustMe (0.00) or RawMean (0.08) due to acquiring a large portion of negative feedback.

This analysis overall provides clear evidence that RawMean is an unfair metric that possibly produces misleading information. RawMean also ignores the change in the community topology over time because it solely calculates the simple average of the gotten evaluation scores. on the contrary, TrustMe sounds to be more logical in producing trust scores that are in harmony with the scores produced by other measures. The evolvement of trust is subject to positive or negative events that occur when actors involved in an interaction/transaction. The trust model should take into consideration that actors are different in terms of being tolerant or intolerant, and this affects the provided judgment score and the trust evolvement as well. Additionally, other factors have an influence on trust evolvement such as the number of relationships, interaction time, the reputation of the interacted parties/actors. All these elements move the trust index up or down based upon the situation. Another critical factor is malicious attacks that exploit some vulnerabilities in the trust model in order to change the trust score. This factor is not taken into account in this case but will be done in future work. To sum up, the TrustMe metric as in traduced above employs several factors to mitigate the risk of bad actors in modifying trust scores. Studying trust evolvement aids in understanding how trust develops and if the applied trust model is effective or not. TrustMe model is promising yet it requires some sub models to be injected or improved to increase its efficacy such as enhancing the punishment mechanism. Trust is a complicated topic that needs more work until innovating and building an effective model that can deal with several situations.

7. SIMULATION EXPERIMENTS AND RESULTS

7.1. Tools and Languages

In these experiments, R was used to compute some factors before proceeding the experiments such as the popularity factor. Whereas C#.NET was used to develop and build the simulation systems, and n user interface was designed to receive the simulations settings.

7.2. Experimental Evaluation of Unshared-Reputation Context

To evaluate the efficiency of the proposed model (TrustMe), four sets of experiments were conducted. Two metrics were used to measure trust evaluation accuracy and the transaction success rate. Additionally, two other approaches (PeerTrust & RawMean) were modeled along with our model for comparison purposes. Each set of experiments made of several variables to evaluate the three models using different settings.

7.2.1. Simulation Setup

Due to the lack of real datasets that represent real trust scores associated with other variables, we implemented, a simulation system that is based upon probability theory using C#.Net and R to generate our datasets. This simulation involves five sub-models: community model, threat model, transaction model, evaluation model, and trust model. Figure 55 shows the abstract design of the simulation system. Additionally, the simulation system consists of a set of variables that form the simulation settings and control the generation process. The values of these variables mostly came from a real dataset. An OTC-bitcoin dataset was used as a reference in the process of generating our datasets. The generated datasets were then used to evaluate the trust models. Table 19 summarizes the simulation settings.

Table 19. The Summary of the Simulation Variables.

Variable	Value	Description
NOA	ref.	The number of actors in the community model based on the reference (OTC-bitcoin).
NOT	ref.	The total number of transactions that will be performed is based on the reference.
NOT. Range	min 1 max ref.	The number of transactions that every actor may perform which could be only one transaction or many.
Generation Mode	Random	The number of transactions that the actor should perform is generated randomly (skewness ≠ 0).
Crr	ref.	Cooperative reciprocal rating: the rating process could be reciprocal between the two parties based on this value.
Uar	Vary	Untrustworthy actors' ratio: this represents the number of untrustworthy actors {0.10, 0.25, 0.50}.
Mrr	Fixed	Malicious reciprocal rating: the rating process could be reciprocal based on this value when a malicious action occurs.
Mbr	Vary	Malicious behavior rate: the malicious behavior could occur based on this value {0.25, 0.50}.

7.2.1.1. Community Model

Before establishing relationships between parties, a community c_k primarily needs to be formed. A community model is an approach to establish an existence around a common purpose where actors are motivated to join and engage in interactions. Therefore, the first step in the simulation process is building a community c_k that consists of a number of actors (*NOA*). Those actors (*A*) can be human beings, entities, devices, software, and so forth. We obtained the number of actors from a real dataset that represents an old bitcoin community called "OTC-bitcoin." The first simulation design starts with a small number of actors based on the given dataset reference (OTC-bitcoin), and then the number grows. The community c_k is divided into two parts trustworthy and untrustworthy actors. The number of untrustworthy actors is captured by applying the variable Uar.

$$C_k \equiv \{A_{trustworthy} \cup A_{untrustworthy}\}$$

$$C_k \equiv \{\{a_1, a_2, \cdots, a_n\} \cup \{a_1, a_2, \cdots, a_m\}\}$$

where C denotes to k community; $A_{trustworthy}$ denotes trustworthy actors whereas $A_{untrustworthy}$ denotes untrustworthy actors.

7.2.1.2. Threat Model

A threat is a term that indicates anything that may cause harm or take advantage of a vulnerability. The purpose of the threat model is to reveal the weakness of the proposed model and also to show how effective this model is in dealing with such threats. In our experiment, the untrustworthy actors are mainly the source of threats when they behave maliciously. When a transaction takes a place, an untrustworthy actor may intentionally behave maliciously and not providing the requested service. Besides that, the untrustworthy actor gives a misleading rating or evaluation to the second party in order to conceal the malicious behavior. The untrustworthy actor not always acts malicious to not be recognized as a source of threats. The rate of recurrence of the malicious behavior for an untrustworthy actor is modeled by Mbr. Also, the overall average of the malicious behavior (MB_{avg}) in the community is acquired by the following equation.

$$MB_{avg} = (Uar * NOA) * Mbr$$

7.2.1.3. Transaction Model

This model allows actors to communicate and interact with each other. This interaction usually causes a change in relationships between those actors. The transactions in our experiments are randomized-based. Actors are chosen randomly to carry out transactions between one another. The number of transactions that one actor can perform is randomly generated and fall between [min, max], and it is attributed to the simulation settings. When a transaction takes a place, a trustworthy actor constantly behaves honestly (cooperative). However, an untrustworthy actor behaves in a dishonest way (malicious) based on the simulation settings (Mbr). Transactions will be performed over time (t), and each transaction will have a timestamp. To be more realistic, a collection of transactions will be carried out simultaneously at time t. This is called transaction waves as a cross-validation mechanism.

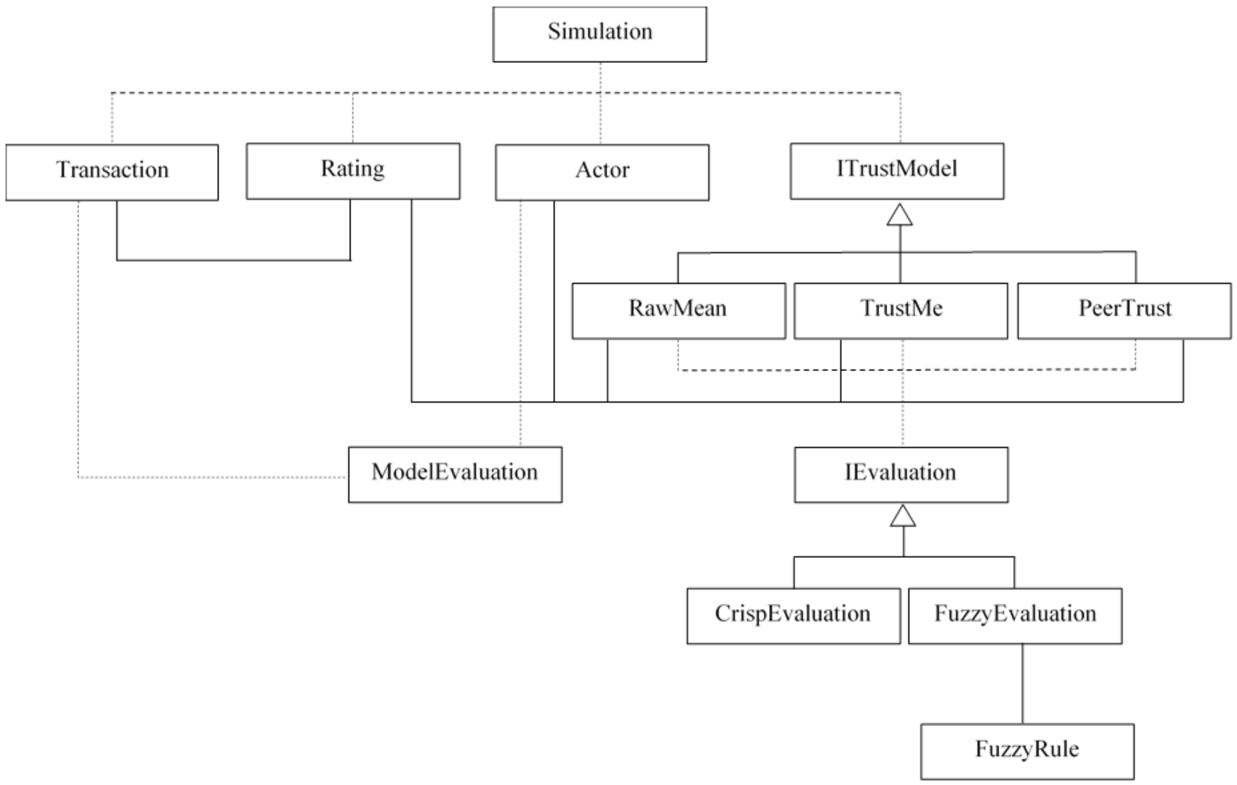


Figure 55. The Abstract Design of the Simulation System.

7.2.1.4. Evaluation Model

An evaluation model simply concerns about specifying the assessment criteria to evaluate actors in a given community. After performing transactions, actors may want to evaluate one another through a mechanism of evaluation. In this experiment, a rating model was implemented to allow actors to evaluate each other depending on their experience. The rating model employs rating values range between [-10, +10] as -10 indicates an ultimate negative experience and +10 indicates an ultimate positive experience. If a trustworthy actor involves in a transaction with an untrustworthy actor resulting in not receiving the requested service, the trustworthy actor may provide a negative rating against the untrustworthy actor.

$$R_{a_i} \equiv \{r_1, r_2, \ldots, r_v\}$$

$$\forall r_i \in R_{a_i} \equiv [-10, +10]$$

7.2.1.5. Trust Model

A trust model comprises of three different trust models (TrustMe, PeerTrust, & RawMean). This component generates trust scores based on each model's design. Each actor will have three different trust scores calculated by employing these trust models. Therefore, one actor may end up with three different judgments. Classifying actors using crisp evaluation (trustworthy or untrustworthy) is based on three different trust models and three different thresholds. Whereas fuzzy evaluation is another evaluation mechanism to cluster actors into five categories as highly trustworthy, trustworthy, suspicious, untrustworthy, and highly untrustworthy. These five clusters generated using thresholds acquired from a real dataset after applying density-based clustering.

To compute trust scores, we need to calculate some factors such as the popularity factor which is acquired from the PageRank algorithm using R. The trust scores generated using three different trust models along with three different thresholds to monitor the efficiency of the three models. Thus, each actor has a set of trust records representing different settings. To transform trust scores into the scope of [0, 1], a unity-based normalization (min-max scaling) is employed. However, unity-based normalization is vulnerable to outlier values. Therefore, we first applied z-score normalization (standardization) to avoid such outliers and to generate a distribution with a mean of 0 and a standard deviation of 1.

7.2.2. Dataset Structure

In this experiment, we create three main entities to mimic the real environment of calculating trust. Actor entity consists of four attributes: actor ID, actor role (trustworthy or untrustworthy), Mbr, and NOT along with actor record as a sub-entity that made of seventeen attributes: actor ID, trust model (TrustMe, PeerTrust, and RawMean), trust score ([0,1]), threshold (0.60, 0.70, & 0.80), judgment (trustworthy or untrustworthy), popularity, number of done transactions, number of rejected transactions, number of malicious transactions, number of non-malicious transactions, PRR, NRR, tolerance rate, punishment rate, reward rate, bias rate, dishonesty rate. Whereas a transaction entity is inclusive of nine attributes: transaction ID, timestamp, transaction activity, transaction status, first party, second party, transaction wave number, trust model, threshold. Rating entity formed from nine attributes: rating ID, source (actor), target (actor), rating score ([-10, 10]), timestamp, time weight, rating weight, source actor popularity, transaction ID. These entities were used to compute trust scores and to store the generated data.

7.2.3. Dataset Generation

During the process of generating our datasets, normal and uniform distribution was applied to procure random data. For instance, the number of transactions that one actor can perform falls between a minimum and maximum, so actors do not have the same number of transactions to perform. It is possible that one actor performs only one transaction, while another actor performs many transactions. These numbers are generated randomly based on a given range. To assign a timestamp to each transaction, a bootstrap resampling was used to pick timestamps from a list of timestamps extracted from the OTC-bitcoin. Each transaction wave takes a list of timestamps as a sample from the original list and then distributed randomly on the transactions of that wave. We identified how many transaction waves for each set of experiments by applying the equation of *NOW*, whereas the maximum number of transactions that one actor can perform is obtained by the equation of *MNOT*. Another important variable is the cooperative reciprocal rating (*Crr*), and its value came from the referenced dataset. We first sliced the dataset into years from 2010 until 2016, and we further compute the reciprocity values for each year using R. The number of actors for each year is considered the number of actors for each simulation design as shown in table 20. The number of transactions for each simulation design is extracted from the number of ratings and the reciprocity value as in the equation of *NOT*. Tables 21, 22, and 23 show samples of the generated and computed data.

$$NOW = NOT^2/NOA^2$$

$$MNOT = \left({}^{NOT}/_{NOA}\right) * NOW$$

$$NOT = \frac{NOR * reciprocity}{2} + NOR - (NOR * reciprocity)$$

where: *NOW* denotes the number of transaction waves; *MNOT* denotes the maximum number of transactions; NOR indicates the number of ratings.

Table 20. Simulation Settings.

Simulation Design	Variables								
	NOA	*NOT*	*NOW*	*NOT Range*	*Generation Mode*	*Crr*	*Uar*	*Mrr*	*Mbr*
Simulation 01	55	91	3	[1-12]	Random	0.72	0.10	0.98	0.25
Simulation 02	1,637	4,424	10	[1-65]	Random	0.88	0.25	0.98	0.25
Simulation 03	3,162	10,139	19	[1-114]	Random	0.83	0.25	0.98	0.25
Simulation 04	5,881	21,532	32	[1-288]	Random	0.79	0.50	0.98	0.50

7.2.4. Trust Evaluation Accuracy (*TEA*)

Since our experiments ended up with classifying actors into trustworthy and untrustworthy, accuracy evaluation metric is utilized to measure the efficiency of the trust models. Accuracy is a popular evaluation metric for classification problems. It is simply the ratio of the correct predictions divided on all predictions (correct and incorrect). This metric reveals the performance of the trust models with the involvement of the number of transactions, the number of trustworthy actors, the number of untrustworthy actors, malicious behavior, timestamps, credibility, and popularity. If a judgment of one actor is equivalent to its assigned role, it is deemed as correct trust evaluation. Otherwise, it is considered an incorrect trust evaluation. This metric manifests at what level we can rely on trust models to make our trust decisions. As we employ the transaction waves principle, it is considered as a cross-validation technique as transactions are divided into groups (folds) to be carried out separately and to generate trust scores for each fold. This also helps to check for overfitting and how our model will be generalized.

$$TEA = {}^{number\ of\ correct\ predections}\!/_{the\ total\ number\ of\ all\ predictions}$$

Table 21. Sample of Actor Data.

ID	Role	Trust.Score	Trust.Model	*Thr.*	Judgment	Popularity	*Mrb*	*NOT*	*MT*
1	Untrustworthy	0.47	PeerTrust	0.80	Untrustworthy	0.011427591	0.5	99	37
2	Untrustworthy	0.66	PeerTrust	0.80	Untrustworthy	0.006705053	0.5	105	46
3	Trustworthy	0.72	PeerTrust	0.80	Untrustworthy	0.015202776	0	74	0
4	Trustworthy	0.99	PeerTrust	0.80	Trustworthy	0.014024468	0	106	0
5	Trustworthy	0.82	PeerTrust	0.80	Trustworthy	0.009913554	0	25	0
6	Untrustworthy	0.80	PeerTrust	0.80	Trustworthy	0.014038401	0.5	56	11
7	Trustworthy	0.50	PeerTrust	0.80	Untrustworthy	0.010072222	0	62	0
8	Trustworthy	0.98	PeerTrust	0.80	Trustworthy	0.009019818	0	134	0
9	Trustworthy	0.72	PeerTrust	0.80	Untrustworthy	0.011581169	0	160	0
10	Trustworthy	0.82	PeerTrust	0.80	Trustworthy	0.010502233	0	31	0
11	Untrustworthy	0.49	PeerTrust	0.80	Untrustworthy	0.018285707	0.5	22	5
12	Untrustworthy	0.48	PeerTrust	0.80	Untrustworthy	0.009495046	0.5	76	20
13	Trustworthy	0.95	PeerTrust	0.80	Trustworthy	0.010642627	0	88	0
14	Trustworthy	0.37	PeerTrust	0.80	Untrustworthy	0.00636896	0	124	0
15	Untrustworthy	0.61	PeerTrust	0.80	Untrustworthy	0.006572858	0.5	70	12
16	Trustworthy	0.54	PeerTrust	0.80	Untrustworthy	0.008949003	0	7	0
17	Trustworthy	0.78	PeerTrust	0.80	Untrustworthy	0.012955984	0	52	0
18	Untrustworthy	0.51	PeerTrust	0.80	Untrustworthy	0.006823997	0.5	73	19
19	Untrustworthy	0.72	PeerTrust	0.80	Untrustworthy	0.008636498	0.5	70	25

7.2.5. Transaction Success Rate (*TSR*)

TSR is calculated by fractioning the number of successful transactions over the total number of all transactions over a given time. A transaction is deemed successful if both parties decided to be cooperative, while a transaction is deemed unsuccessful if one or both parties decided to act malicious. The presence of maliciousness means the security level of the community will drop down and means the higher the number of malicious transactions is, the lower the security level is, and vice versa. This metric additionally gives an elucidation of the productivity level of a community. It implies that when the number of cooperative transactions is higher, the productivity level will be higher. We compare *TSR* of our model with *TSRs* of PeerTrust and RawMean models. The assumption is that the absence of an efficient trust instrument steers us toward a community with low TSR as actors cannot make proper trust decisions, and consequently, they will be victims of the risk of untrustworthy actors.

$$TSR = {}^{number\ of\ successful\ tranactions}\!/_{the\ total\ number\ of\ all\ transactions}$$

Table 22. Sample of Transaction Data.

Trans.ID	Timestamp	First.Party	Second.Party	Trust.Model	*Thr.*	Status	Activity
1	1291591212	4682	2474	TrustMe	0.70	Deal	Cooperative
2	1291591212	67	217	TrustMe	0.70	Deal	Cooperative
3	1291591212	2080	5251	TrustMe	0.70	Deal	Cooperative
4	1291591212	5100	1923	TrustMe	0.70	Deal	Cooperative
5	1291578872	3656	5315	TrustMe	0.70	Deal	Cooperative
6	1291578872	5866	5881	TrustMe	0.70	Deal	Cooperative
7	1291578872	3883	1634	TrustMe	0.70	Deal	Cooperative
8	1291578872	2432	434	TrustMe	0.70	Deal	Malicious
9	1291578872	5816	4947	TrustMe	0.70	Deal	Cooperative
10	1291684580	1825	2768	TrustMe	0.70	Deal	Cooperative
11	1291684580	4976	3573	TrustMe	0.70	Deal	Cooperative
12	1291684580	165	400	TrustMe	0.70	Deal	Cooperative
13	1291684580	1613	5292	TrustMe	0.70	Deal	Cooperative
14	1291684580	3207	5554	TrustMe	0.70	Deal	Cooperative
15	1291684580	4999	4092	TrustMe	0.70	Deal	Cooperative

Table 23. Sample of Rating Data.

ID	Source	Source.Role	Target	Target.Role	Rating.Score	Activity	Model	*Thr.*
32403	5312	Trustworthy	5365	Trustworthy	3	Cooperative	RawMea	0.70
32404	5365	Trustworthy	5312	Trustworthy	9	Cooperative	RawMea	0.70
32405	1432	Untrustwort-	2922	Untrustworthy	2	Cooperative	RawMea	0.70
32406	2922	Untrustwort-	1432	Untrustworthy	1	Cooperative	RawMea	0.70
32407	4152	Untrustwort-	2336	Untrustworthy	- 4	Malicious	RawMea	0.70
32408	2615	Untrustwort-	261	Trustworthy	-2	Malicious	RawMea	0.70
32409	261	Trustworthy	2615	Untrustworthy	-7	Malicious	RawMea	0.70
32410	4954	Trustworthy	3336	Untrustworthy	6	Cooperative	RawMea	0.70
32411	3171	Untrustwort-	3763	Trustworthy	8	Cooperative	RawMea	0.70
32412	2340	Trustworthy	4758	Untrustworthy	-5	Malicious	RawMea	0.70

7.2.6. Simulation Analysis

In this section, a set of four simulation designs are described and characterized. Each simulation design has different settings to imitate various scenarios. The first design starts with a small number of actors, a small number of relationships, and a low rate of malicious attacks. Subsequent simulation designs increase these numbers to monitor the performance and efficiency of the models under study various settings.

7.2.6.1. Simulation Design 01

In this design, the number of actors sets to be 55 (*NOA = 55*). The number of transactions that to carry out is 91 (*NOT = 91*). The number of transaction waves set to be 4 (*NOW = 4*), so the total number of transactions is divided into 4 groups. The number of transactions that one actor can perform is vary based on the specified range ($NOT_{range} = [1, 12]$). Hence, the number of transactions for every actor to perform falls between 1 and 12 and randomly assigned. According to the analysis of the referenced dataset, the cooperative reciprocity rate set to be 0.72 (*Crr = 0.72*). Actors are not required to deliver ratings for each other. This implies that the evaluation could be in a two-way form (both evaluate each other) or in a one-way form (only one actor evaluates the other). The percentage of untrustworthy actors in the community sets to be 0.10 (*Uar = 0.10*). Thereby the number of untrustworthy actors is only 6 actors with this setting. The malicious behavior factor in this experiment is determined by the malicious behavior rate which sets to be 0.25 (*Mbr = 0.25*). Thus, a quarter of the transactions of the untrustworthy actors will be malicious.

After running the simulation system, the candidate actors are picked randomly to carry out transactions with each other. The transactions are performed over 4 waves, and before the start of a wave, trust scores of actors are computed. These trust scores are employed to classify actors whether they are trustworthy or untrustworthy. The decision is made based on the score of trust and a crisp evaluation using three different thresholds (0.60, 0.70, & 0.80). An actor is judged as trustworthy if its trust score is equal or greater than the threshold ($TS \geq threshold \rightarrow trustworthy$) whereas an actor is judged as untrustworthy if its trust score is less than the threshold ($TS < threshold \rightarrow trustworthy$). We then measure the accuracy of each transactions' wave and we compute how much maliciousness in that wave to eventually curve these values to see how trust models perform with the movement of malicious behavior. We further compute the transaction success rate to evaluate the security and productivity of the given community.

- ***Simulation Results***

Figures 56-61 depict the performance of the three trust models using different settings. The Three graphs illustrate trust evaluation accuracy concerning malicious behaviors, and the other three graphs illustrate transaction success rates of all models. It is discernible that all models (TrustMe, PeerTrust, and RawMean) seem to be effective specifically when the malicious behavior rate is at low levels. Even after increasing the rate of malicious behavior, the performance of all models remains at high levels. When using a lower threshold ($T_{threshold} = 0.60$), the accuracy values are steadier and are around the nineties. This implies that all models are considered highly accurate in evaluating actors. Despite RawMean achieves good performance at higher thresholds ($T_{threshold} = 0.70 \; or \; 0.80$), its accuracy is the lowest compared to TrustMe and PeerTrust. It sounds that TrustMe and PeerTrust are competing with each other, and it can be said that their performance and accuracy are equivalent. Succinctly, the number of actors in this community is very small, and even the number of malicious actors is only 10%($Uar : 0.10 \equiv 6 \; untrustworthy \; actors$) of the entire community. Therefore, a solid conclusion cannot be built from this experiment. However, it can be observed that RawMean underperforms TrustMe and PeerTrust as its accuracy declines when the malicious behavior rate increases.

Table 24. Means of Trust Evaluation Accuracy and Transaction Success Rate (SD01).

Threshold	TrustMe		PeerTrust		RawMean	
	Mean of TEA	*Mean of TSR*	*Mean of TEA*	*Mean of TSR*	*Mean of TEA*	*Mean of TSR*
0.60	0.97	0.94	0.97	0.94	0.94	0.86
0.70	0.97	0.97	0.97	0.97	0.92	0.92
0.80	0.96	0.97	0.95	0.96	0.90	0.88

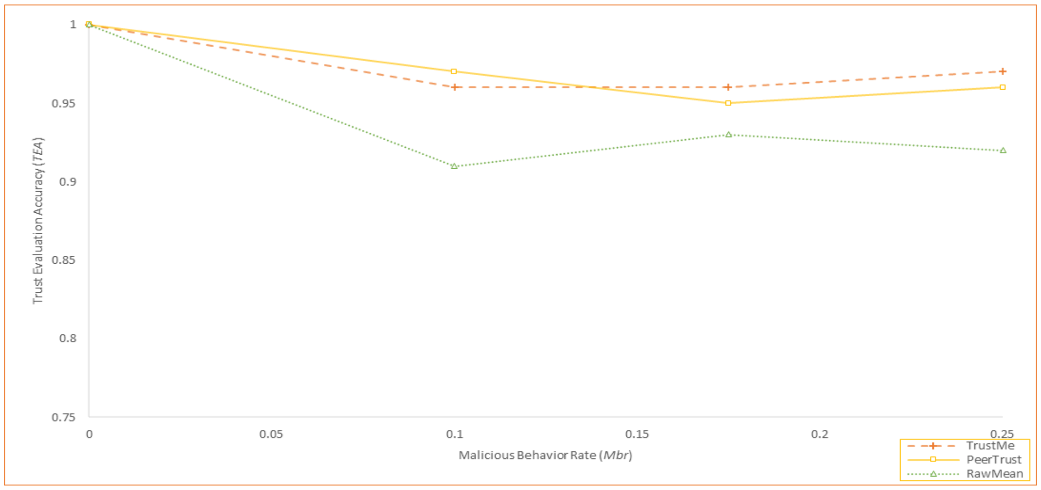


Figure 56. Trust Accuracy Evaluations for all Models with $T_{threshold}$ = *0.60* (SD01).

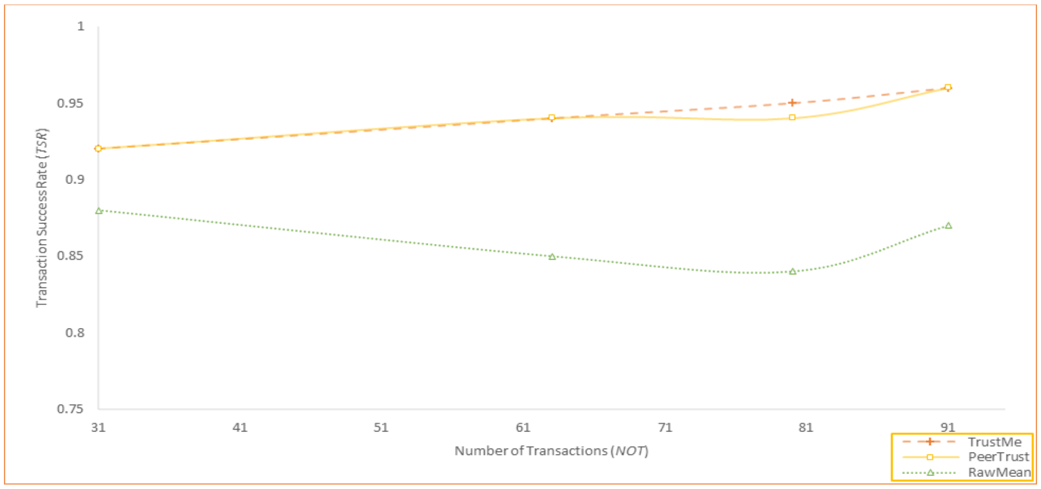


Figure 57. Transaction Success Rates for all Models with $T_{threshold}$ = *0.60* (SD01).

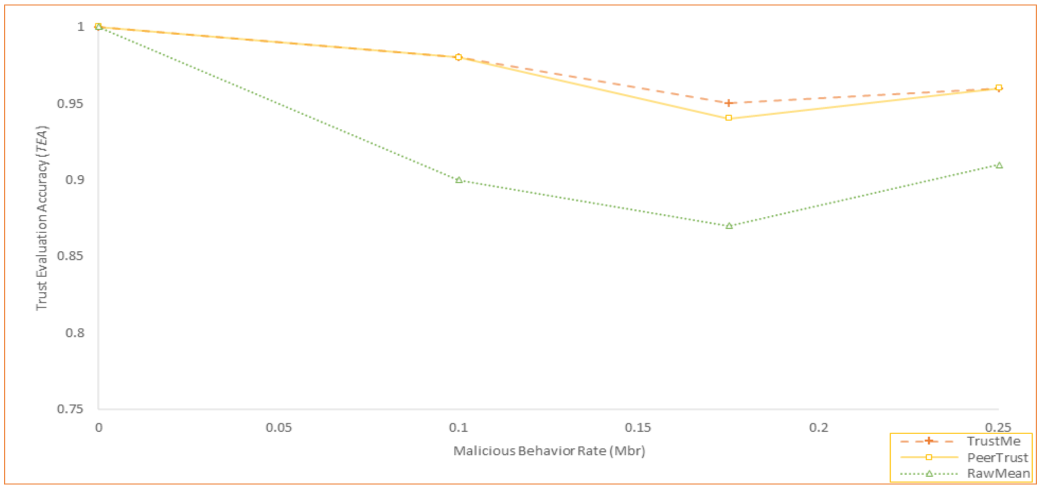


Figure 58. Trust Accuracy Evaluations for all Models with $T_{threshold}$ = *0.70* (SD01).

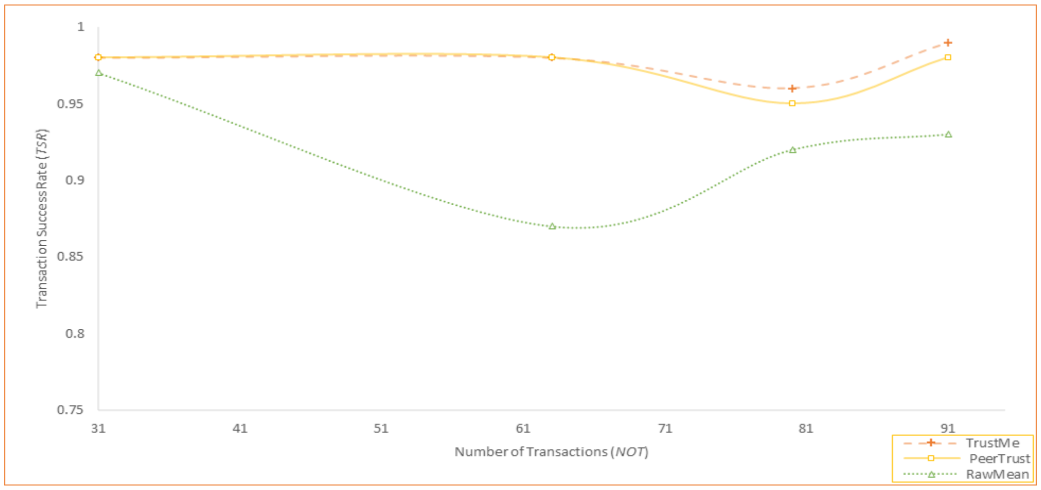


Figure 59. Transaction Success Rates for all Models with $T_{threshold}$ = *0.70* (SD01).

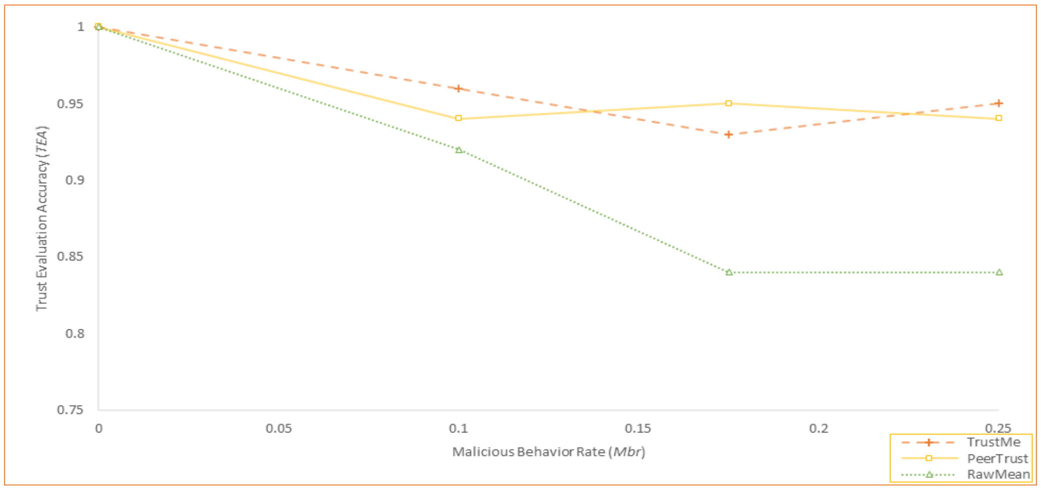


Figure 60. Trust Accuracy Evaluations for all Models with $T_{threshold}$ = *0.80* (SD01).

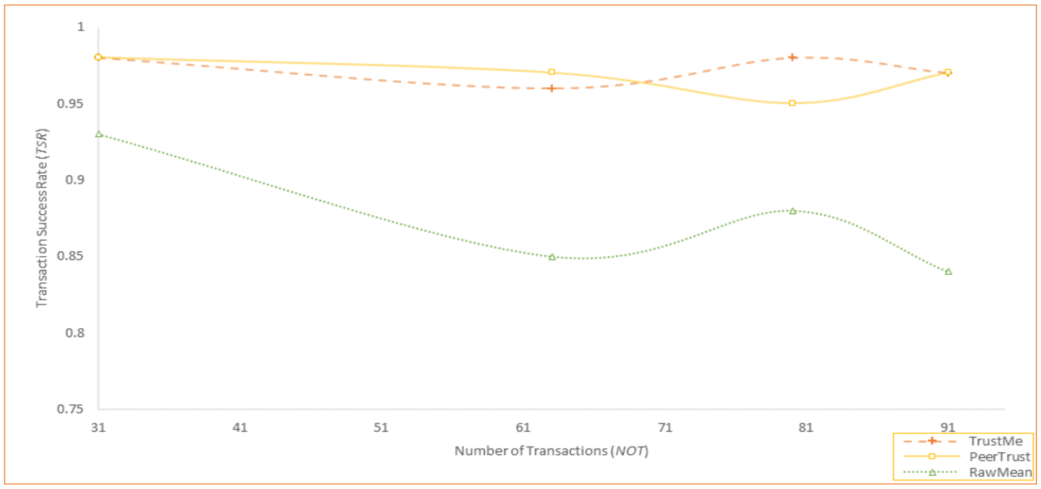


Figure 61. Transaction Success Rates for all Models with $T_{threshold}$ = *0.80* (SD01).

Transaction success rate charts concerning the number of transactions at a certain time show that all communities earn high success rates, but RawMean is less effective than other models. Another observation is that the *TSRs* of all models are quite stable. This indicates all communities are remarkably secure and productive since actors tend to cooperate and be honest rather behaving malicious. This could be ascribed to the small number of untrustworthy actors, and consequently, the number of malicious transactions will likewise be small. Table 24 summarizes the averages of *TEAs* and *TSRs* for all models using three thresholds to ease the comparison process.

7.2.6.2. Simulation Design 02

In this design, the number of actors sets to be 1637 ($NOA = 1637$). The number of transactions to carry out is 4424 ($NOT = 4424$). The number of transaction waves set to be 13 ($NOW = 13$), so the total number of transactions is divided into 13 groups. The number of transactions that one actor can perform is vary based on the specified range ($NOT_{range} = [1, 65]$). Hence, the number of transactions for every actor to perform falls between 1 and 65 and randomly assigned. According to the analysis of the reference dataset, the cooperative reciprocity rate set to be 0.88 ($Crr = 0.88$). The percentage of untrustworthy actors in the community sets to be 0.25 ($Uar = 0.25$). Thereby the number of untrustworthy actors is 410 actors with this setting. The malicious behavior factor in this experiment is determined by the malicious behavior rate which sets to be 0.25 ($Mbr = 0.25$). Thus, a quarter of the transactions of the untrustworthy actors will be malicious.

- ***Simulation Results***

Figures 62-67, as in the previous experiment, depict the performance of the three trust models using different settings. The graph consists of two types of charts representing trust evaluation accuracy and transaction success rate. One interesting observation is that all models perform equivalently when the malicious behavior rate is marginal ($Mbr < 0.05$). Accordingly, the number of malicious transactions will be too small. It can be discovered that the RawMean method works ideally when the number of trustworthy actors is significantly larger than the number of untrustworthy actors. This implies that the number of honest evaluations is capable to nullify the influence of dishonest evaluations. When the malicious behavior rate grows, the *TEA* scores start to decline at the beginning of all approaches. Subsequently, *TEAs* of TrustMe and PeerTrust commence to recover and increase and to be more stable in all settings, while the *TEA* of RawMean continues to decrease dramatically. Table 25 summaries the averages of *TEAs* of the three models using the three thresholds ($T_{threshold} = 0.60, 0.70, or\ 0.80$). It can be seen that in all settings TrustMe outperforms other models (PeerTrust and RawMean), and it seems that RawMean is out of the competition due to its low performance. This is because RawMean is merely relying on the number of transactions and the evaluation score as it aggregates the evaluation scores and eventually finds the average. Whereas TrustMe and PeerTrust are factored models which means that each model has factors aiding in lessening the deception and dishonesty levels.

Figures 63, 65, and 67 also show the transaction success rate for all models in three settings over time. It clear that the RawMean approach is underachieving comparing to TrustMe and PeerTrust approaches. This seems to be aligned with the conclusion of the experiment above of *TEAs.* Another exploration is that *TSRs* of TrustMe and PeerTrust became stable and steady after a slight change at the beginning and their *TSR* scores grow slightly over time. On the first hand, this refers to communities that employ such models to be more secure and productive. It further signifies that actors successfully transact with trustworthy actors and able to acquire reliable information to avoid dealing with untrustworthy actors. On the second hand, *TSR* of the RawMean approach seems to be wavy before became stable at the end but at low levels. This signifies that actors tend to transact with untrustworthy actors and be vulnerable to malicious attacks since they use unreliable information given by a substandard approach. Table 25 epitomizes the averages of *TSRs* of all models for all settings which proves that the TrustMe model is more efficient than other models.

Table 25. Means of Trust Evaluation Accuracy and Transaction Success Rate (SD02).

Threshold	TrustMe		PeerTrust		RawMean	
	Mean of TEA	*Mean of TSR*	*Mean of TEA*	*Mean of TSR*	*Mean of TEA*	*Mean of TSR*
0.60	0.93	0.88	0.90	0.86	0.77	0.68
0.70	0.91	0.89	0.89	0.87	0.78	0.72
0.80	0.88	0.86	0.85	0.83	0.74	0.73

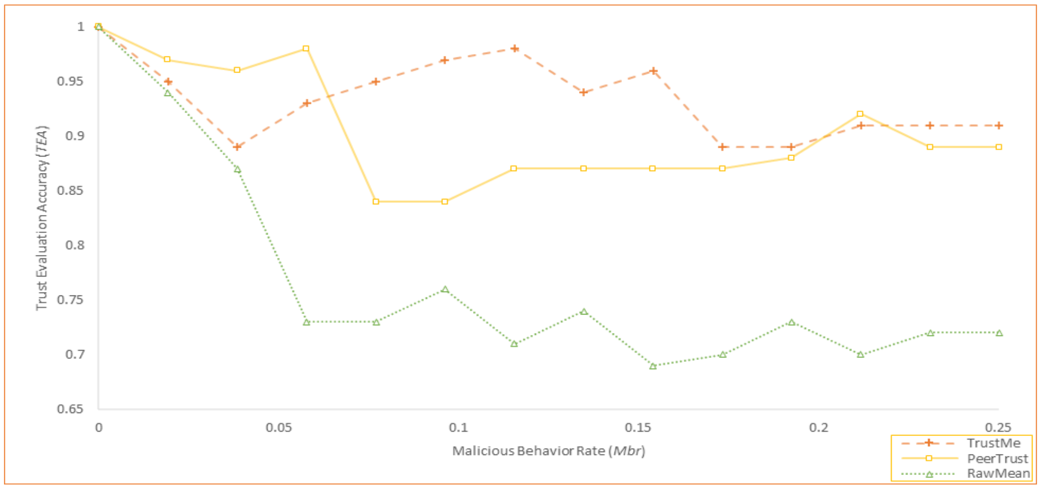


Figure 62. Trust Accuracy Evaluations for all Models with $T_{threshold}$ = *0.60* (SD02).

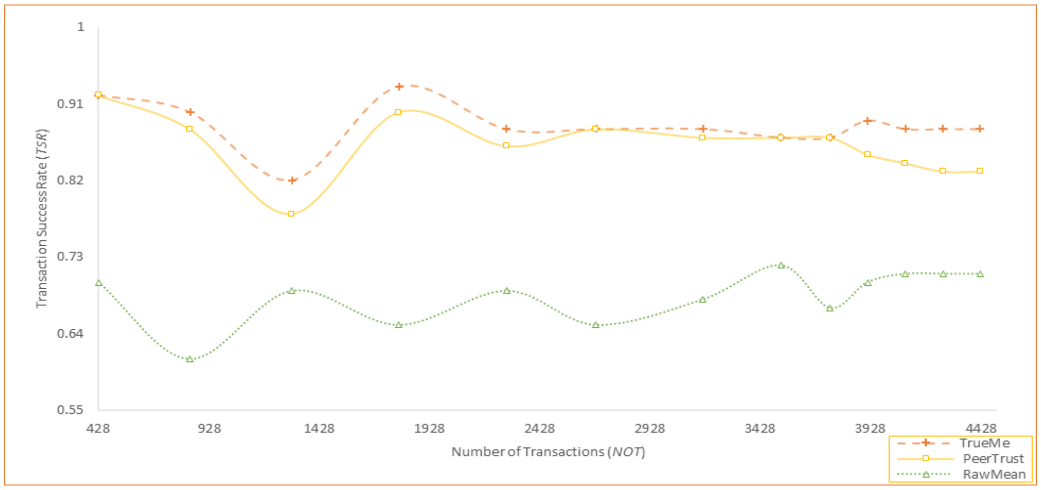


Figure 63. Transaction Success Rates for all Models with $T_{threshold}$ = *0.60* (SD02).

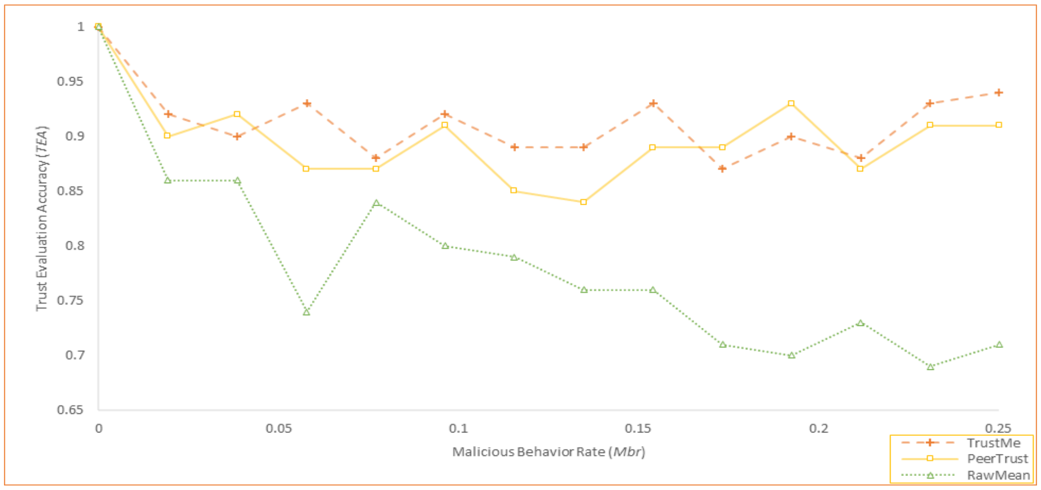


Figure 64. Trust Accuracy Evaluations for all Models with $T_{threshold}$ = *0.70* (SD02).

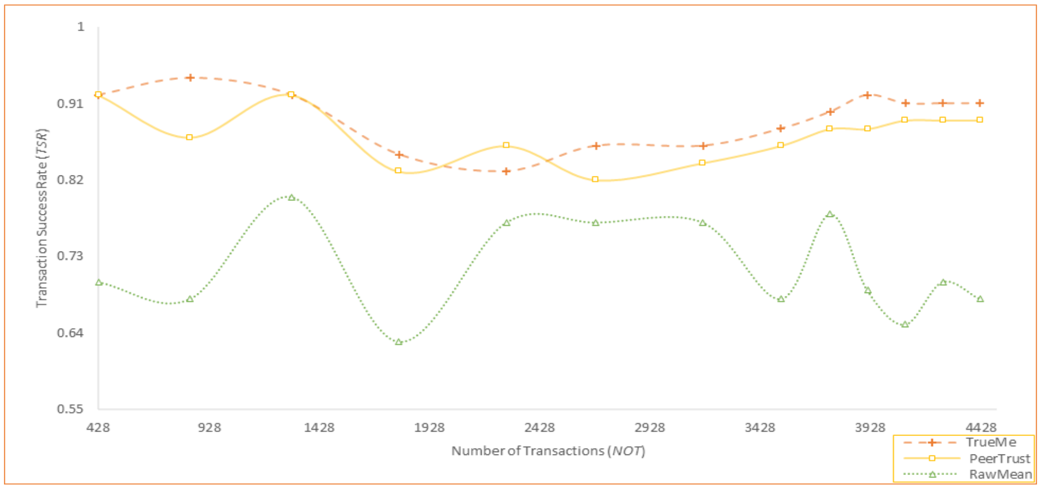


Figure 65. Transaction Success Rates for all Models with $T_{threshold}$ = *0.70* (SD02).

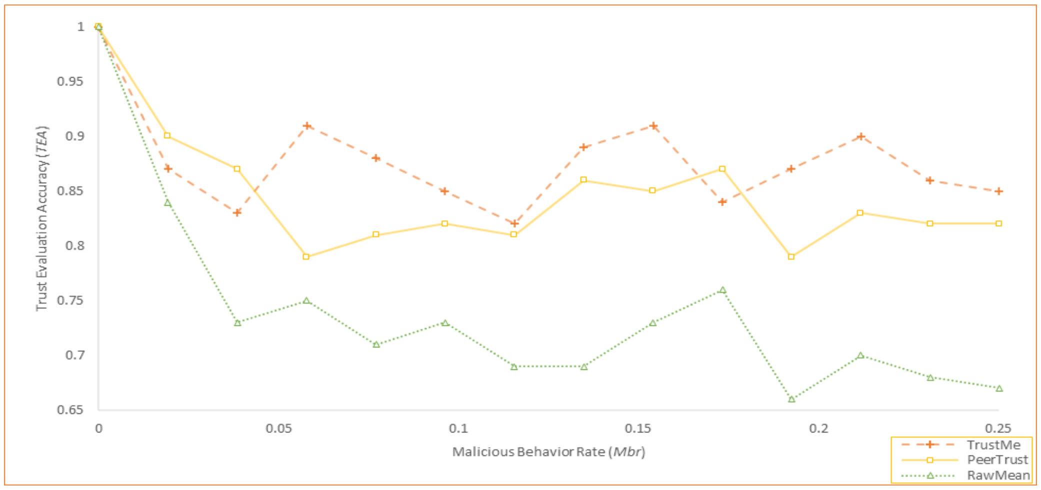


Figure 66. Trust Accuracy Evaluations for all Models with $T_{threshold}$ = *0.80* (SD02).

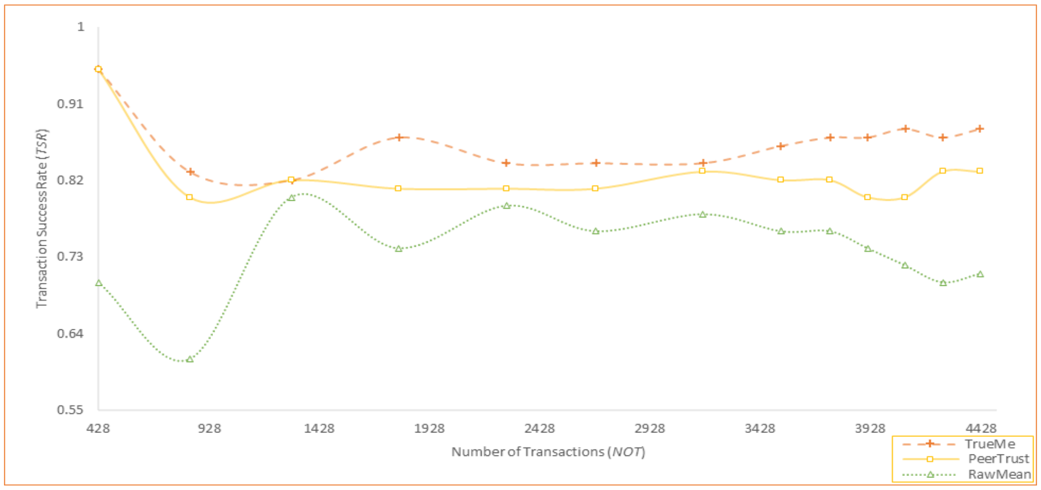


Figure 67. Transaction Success Rates for all Models with $T_{threshold}$ = *0.80* (SD02).

7.2.6.3. Simulation Design 03

In this design, the number of actors sets to be 3162 ($NOA = 3162$). The number of transactions that to carry out is 10139 ($NOT = 10139$). The number of transaction waves set to be 19 ($NOW = 19$), so the total number of transactions is divided into 19 groups. The number of transactions that one actor can perform is vary based on the specified range ($NOT_{range} = [1{,}114]$). Hence, the number of transactions for every actor to perform falls between 1 and 114 and randomly assigned. According to the analysis of the reference dataset, the cooperative reciprocity rate set to be 0.83 ($Crr = 0.83$). The percentage of untrustworthy actors in the community sets to be 0.25 ($Uar = 0.25$). Thereby the number of untrustworthy actors is 791 actors with this setting. The malicious behavior factor in this experiment is determined by the malicious behavior rate which sets to be 0.25 ($Mbr = 0.25$). Thus, a quarter of the transactions of the untrustworthy actors will be malicious.

- ***Simulation Results***

The result of this experiment works as the first affirmation of the previous experiment (Design 2). Figures 68, 70, and 72 represent trust evaluation accuracy and transaction success rate for all models. An interesting exploration is that the *TEAs* of all models still perform equally at the beginning when the *Mbr* is small, and the *TEA* of RawMean is a bit lower than other *TEAs*. Besides, the scores of RawMean starts to continuously drop down when the *Mbr* increases. This is another evidence that RawMean operates exemplarily when the *Mbr* is low. It can also be seen that TEAs of TrustMe and PeerTrust go up and down until they became stable eventually at good levels with a preference given to TrustMe. Table 26 shows that TrustMe significantly outperforms RawMean and slightly outperforms PeerTrust.

Table 26. Means of Trust Evaluation Accuracy and Transaction Success Rate (SD03).

Threshold	TrustMe		PeerTrust		RawMean	
	Mean of TEA	*Mean of TSR*	*Mean of TEA*	*Mean of TSR*	*Mean of TEA*	*Mean of TSR*
0.60	0.93	0.86	0.90	0.83	0.79	0.69
0.70	0.92	0.83	0.89	0.81	0.79	0.69
0.80	0.92	0.82	0.89	0.80	0.76	0.68

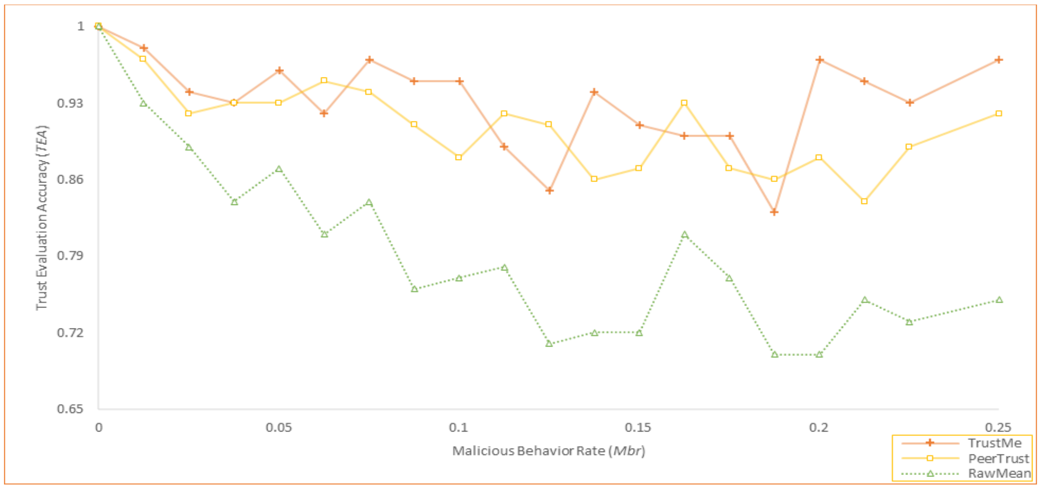


Figure 68. Trust Accuracy Evaluations for all Models with $T_{threshold} = 0.60$ (SD03).

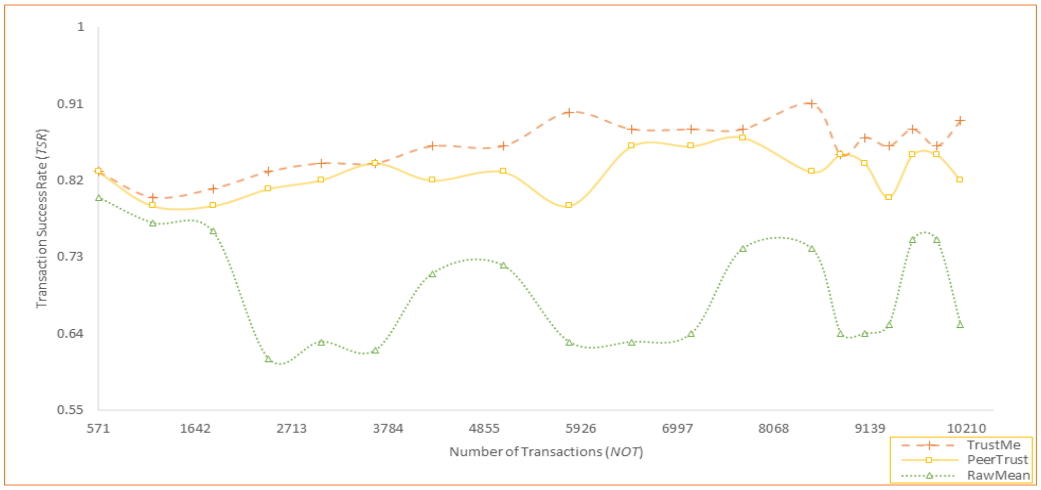


Figure 69. Transaction Success Rates for all Models with $T_{threshold} = 0.60$ (SD03).

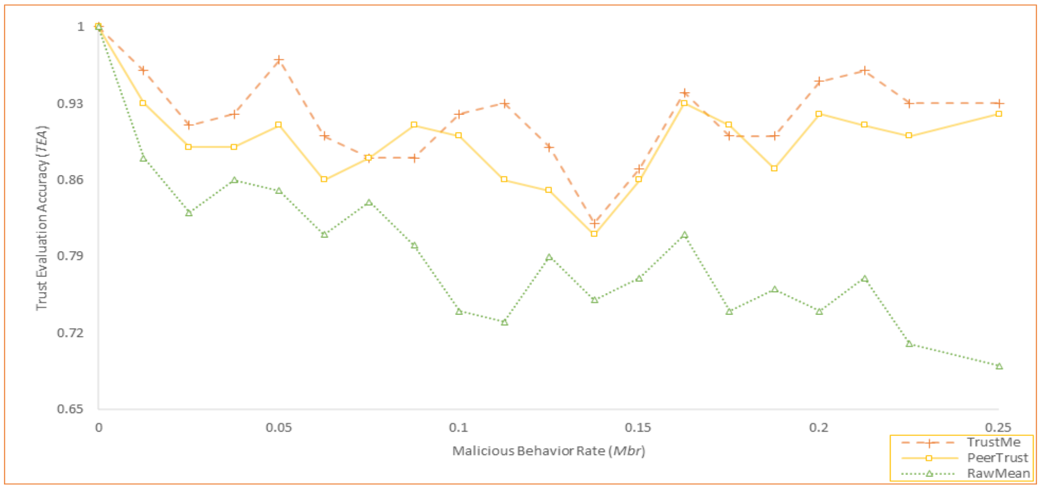


Figure 70. Trust Accuracy Evaluations for all Models with $T_{threshold}$ = *0.70* (SD03).

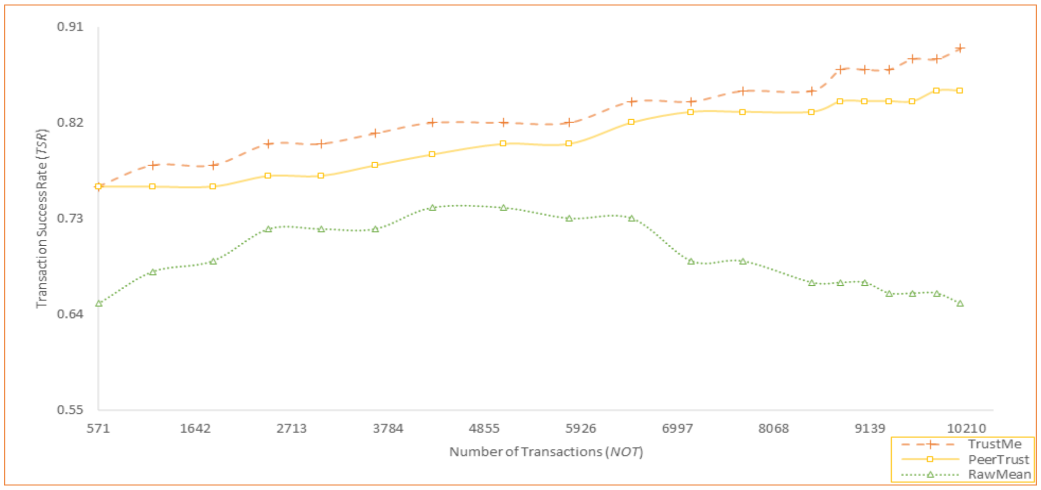


Figure 71. Transaction Success Rates for all Models with $T_{threshold}$ = *0.70* (SD03).

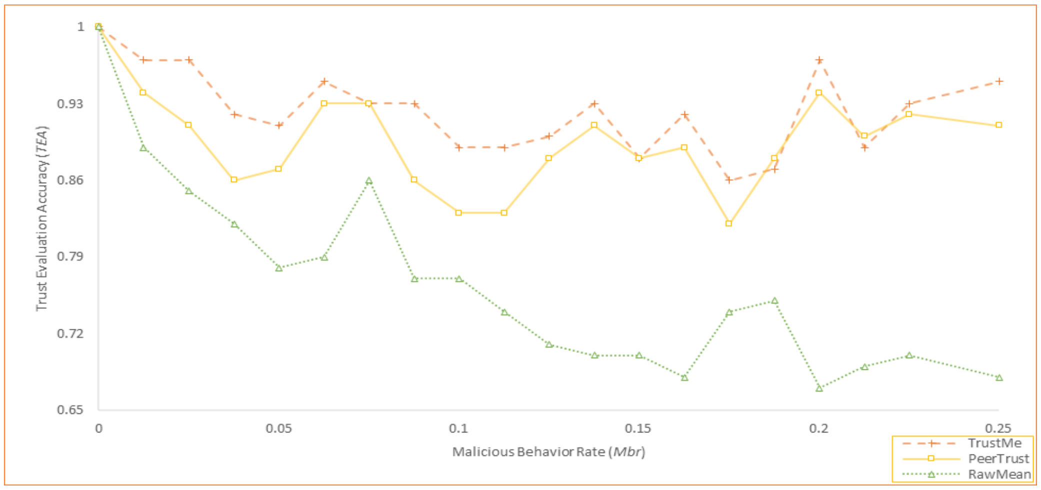


Figure 72. Trust Accuracy Evaluations for all Models with $T_{threshold}$ = *0.80* (SD03).

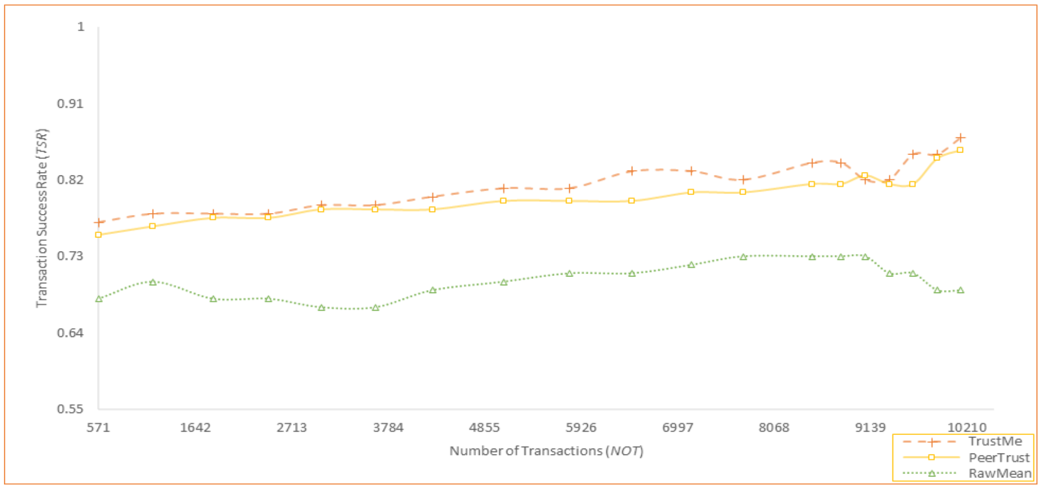


Figure 73. Transaction Success Rates for all Models with $T_{threshold} = 0.80$ (SD03).

Moreover, the transaction success rate for all models in figures 69, 71, and 73 show that the RawMean approach is again less secure and less productive compared to TrustMe and PeerTrust approaches. *TSRs* of TrustMe and PeerTrust increase over time when the number of transactions increases which implies that with more knowledge, these models perform better. This also signifies that with more knowledge, the quality of the given information will be high which increases security and productivity levels. Table 26 furthermore summarizes the average of *TSRs* of all models which indicates that TrustMe outperforms other models.

To sum up, this experiment emphasizes the results procured from the previous experiment. They both prove that RawMean is not effective than the other two models as it performs poorly when malicious behavior increases in the community. Whereas TrustMe outperforms all other models in the context of trust evaluation accuracy and transaction success rate. PeerTrust is still considered effective but below the TrustMe model. The last design (Design 4) is built with a larger community and higher *Mbr* and *Uar* as a second confirmation of this conclusion.

7.2.6.4. Simulation Design 04

In this design, the number of actors sets to be 5881 ($NOA = 5881$). The number of transactions that to carry out is 21532 ($NOT = 21532$). The number of transaction waves set to be 32 ($NOW = 32$). So, the total number of transactions is divided into 32 groups. The number of transactions that one actor can perform is vary based on the specified range ($NOT_{range} = [1{,}288]$). Hence, the number of transactions for every actor to perform falls between 1 and 288 and randomly assigned. According to the analysis of the reference dataset, the cooperative reciprocity rate set to be 0.79 ($Crr = 0.79$). The percentage of untrustworthy actors in the community sets to be 0.50 ($Uar = 0.50$). Thereby the number of untrustworthy actors is 2941 actors with this setting. The malicious behavior factor in this experiment is determined by the malicious behavior rate which sets to be 0.50 ($Mbr = 0.50$). Thus, half of the transactions of the untrustworthy actors will be malicious.

- ***Simulation Results***

The result of this experiment serves as the second affirmation of the experiment of design 2 and the first affirmation of the experiment of design 3. Figures 74-79 consist of two types of charts representing trust evaluation accuracy and transaction success rate for all models. The number of untrustworthy actors in this experiment represents half of the community with $Mbr = 0.50$ which forms a large fraction of malicious transactions that will be carried out. One repeated and evident observation is that TrustMe and PeerTrust achieve good performance with a moderate decrease compared with previous experiments even with this number of untrustworthy actors. This is because the proportion of malicious attack is high which assists in identifying such untrustworthy actors, and as a result, trustworthy actors can avoid dealing with them. Another apparent observation is that the RawMean model still performs deficiently with a dramatic deterioration specifically when *Mbr* increases. Table 27 recapitulates the averages of *TEAs* of all models using three different thresholds. Based on these averages, RawMean is the worst, and TrustMe is the best from a performance perspective.

Figures 75, 77, and 79 also show that *TSRs* of all models are descended and afterward became quite stable when the number of transactions increases. However, *TSR* of RawMean receives a significant drop than others, and it seems not stable. This confirms the previous interpretations as TrustMe and PeerTrust are more secure and productive. RawMean is still vulnerable to such severe attacks and not secure.

Table 27. Means of Trust Evaluation Accuracy and Transaction Success Rate (SD04).

Threshold	TrustMe		PeerTrust		RawMean	
	Mean of TEA	*Mean of TSR*	*Mean of TEA*	*Mean of TSR*	*Mean of TEA*	*Mean of TSR*
0.60	0.84	0.82	0.82	0.75	0.72	0.56
0.70	0.86	0.82	0.83	0.80	0.72	0.59
0.80	0.87	0.80	0.84	0.78	0.71	0.62

To conclude this section, there is a direct positive relationship between *TEA* and *TSR* that can be observed through all experiments. When *TEA* is high which implies the model is more accurate in its evaluation, *TSR* will be high as well. This means actors will receive accurate information, and they will likely select trustworthy actors to transact with averting transacting with untrustworthy actors. In all experiments, we conduct non-collusive attacks that show that TrustMe achieves the highest values followed by PeerTrust and RawMean is the worst.

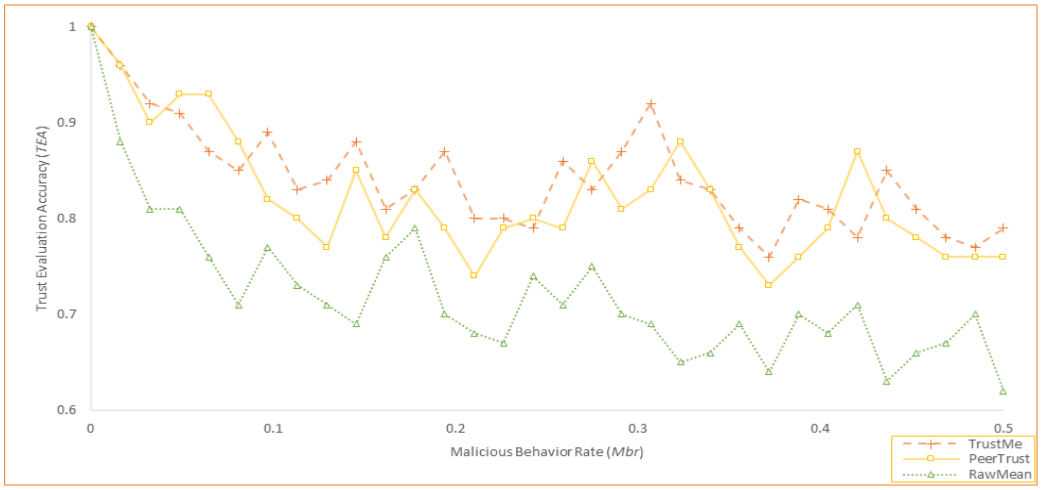


Figure 74. Trust Accuracy Evaluations for all Models with $T_{threshold} = 0.60$ (SD04).

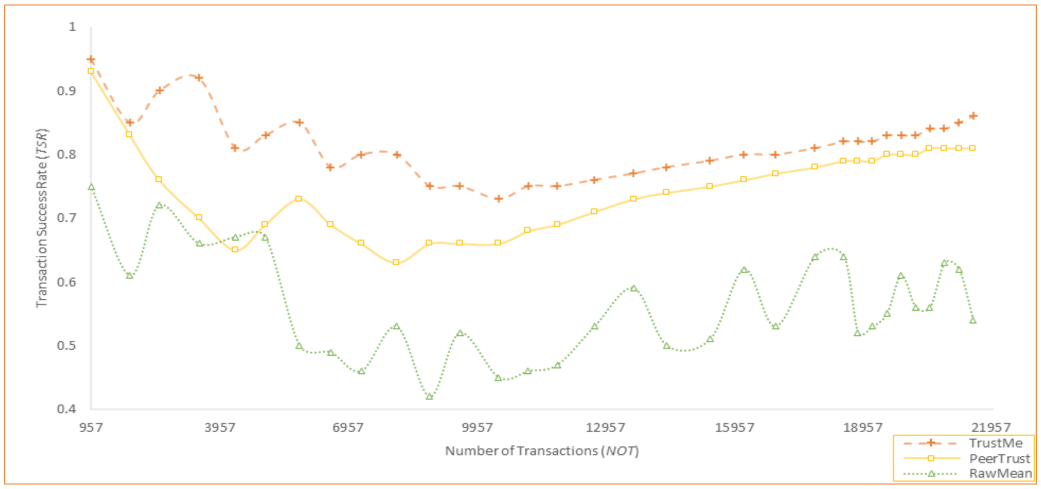


Figure 75. Transaction Success Rates for all Models with $T_{threshold} = 0.60$ (SD04).

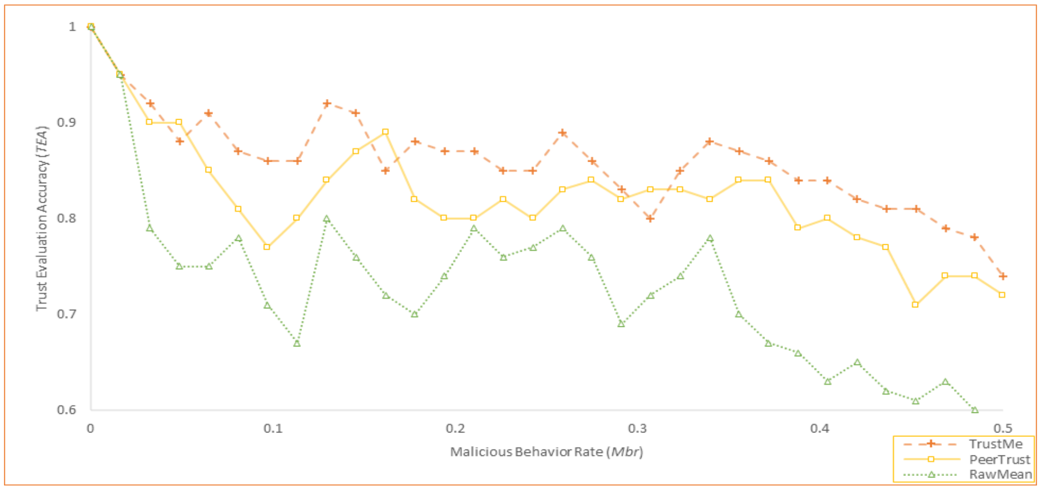


Figure 76. Trust Accuracy Evaluations for all Models with $T_{threshold} = 0.70$ (SD04).

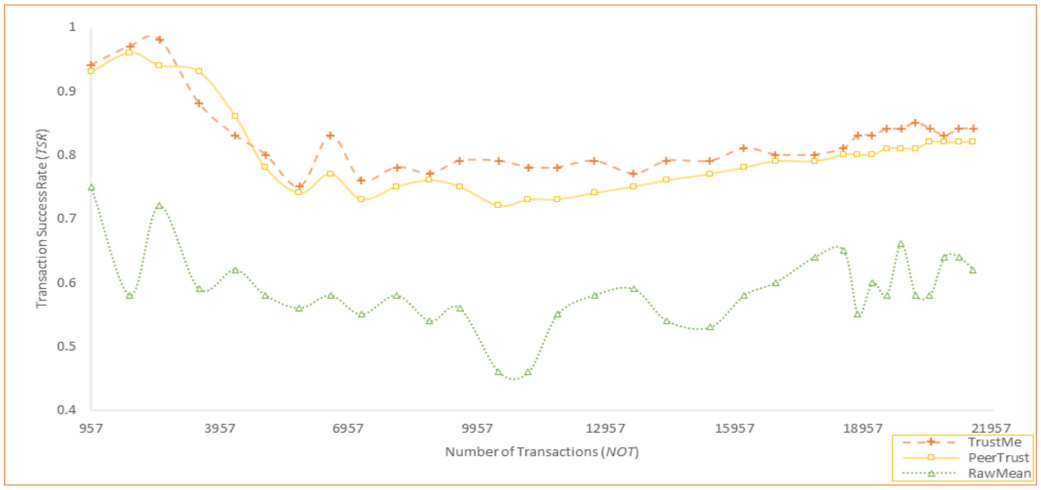


Figure 77. Transaction Success Rates for all Models with $T_{threshold} = 0.70$ (SD04).

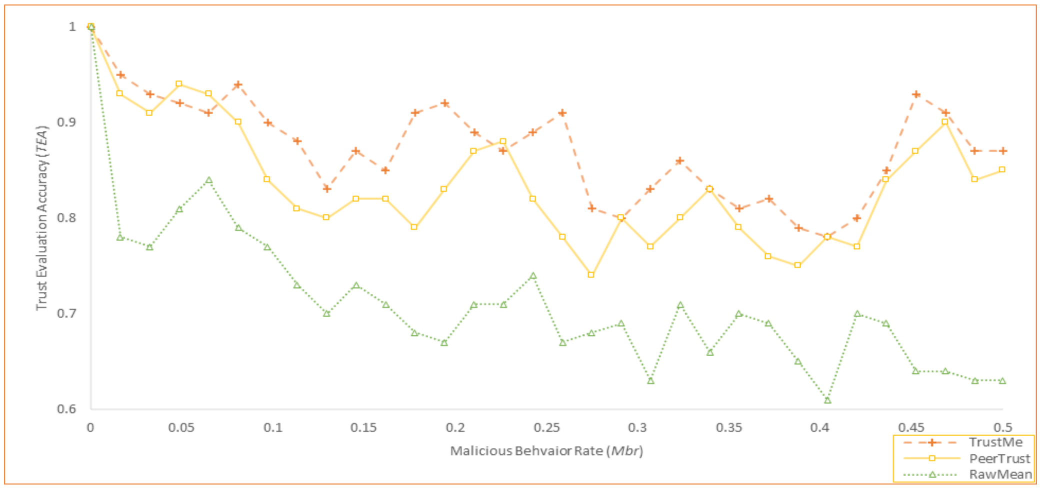


Figure 78. Trust Accuracy Evaluations for all Models with T$_{threshold}$ = *0.80* (SD04).

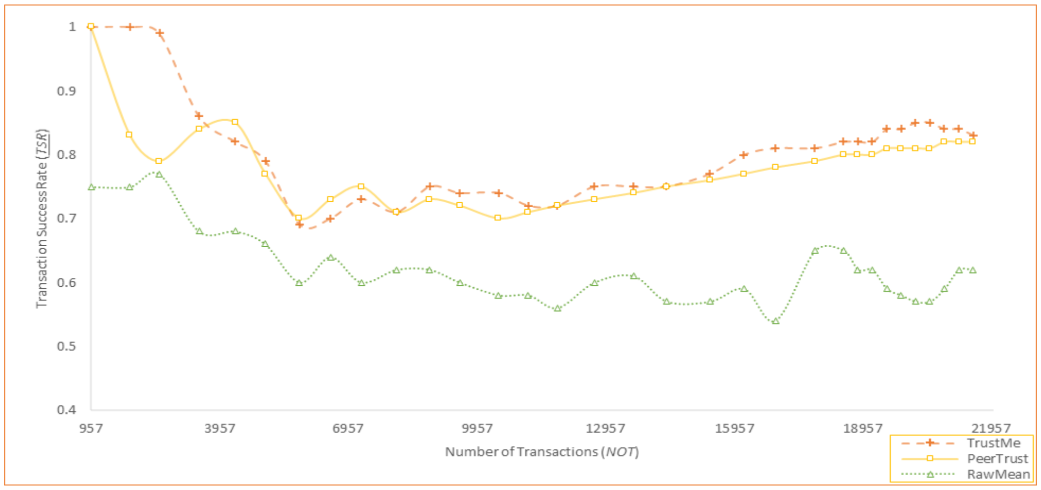


Figure 79. Transaction Success Rates for all Models with T$_{threshold}$ = *0.80* (SD04).

7.3. Simulation Experiment of Blockchain-Based Trust Model

In this context, we aim to see the efficiency of applying cross-community trust using blockchain technology. The design of this experiment involves an adversarial model and a collection of separate communities with their members (actors). Each community has its evaluation mechanism which eventually needs to be standardized. All models (community model, threat model, transaction model, evaluation model, and trust model) that employed in the previous simulations are used. In this experiment, we conduct a collusive attack to see how the trust models perform in cases of this kind. This form of attack differs from the attacks in the previous experiments which are considered as non-collusive. Figure 80 simplifies the design of blockchain Simulation for Cross-Community Trust.

7.3.1. Blockchain Model

This model in the experiment is simple and not as complicated as in our design. This model aims to manage received information from several communities and store them in blocks via Trust-DApp. When one community deliver new data, Trust-DApp will pass these data into the management component to find an actor whose an address that matches the address sent with the new data. In our experiment, we used actor Ids as addresses for the sake of simplicity. When the system finds the actor, the data will be placed on a block and connected to this actor. Trust-DApp is supposed to standardize the received data before storing it. For instance, the popularity factor is different from one community to another. Thus, we receive the size of the community as a factor to minimize or maximize the value of popularity. Another issue that the model handles is the issue of timing. How can we weigh the time after a long time of transacting? The model builds kind of indexing for weighing time. When a new time comes, this indexing is updated. Then, the model starts calculating the trust score of the actor who receives new data and adds a new record for the new trust score.

7.3.2. Adversarial Model

The adversarial model is mainly a sub-model of the threat model. The purpose of this model is to mimic a malicious attack targeting an actor in a specific community and to manifest how trust models perform with such an attack. Adversarial patterns will be in a form of values injected into trust models. Attackers deliberately designed such inputs to make trust models providing misleading information. In this experiment, a malicious adversarial model involving conspiracy is designed to deceive the community. This implies that any actor or a group of actors may collude secretly to execute a pernicious plan. Further, the attack may take the form of a Sybil attack where the reputation of the community members is dishonestly diverted by creating multiple identities.

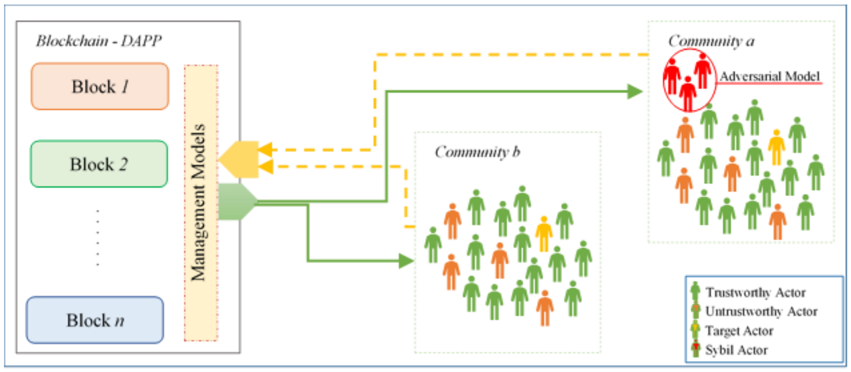


Figure 80. Blockchain Simulation Design for Cross-Community Trust.

7.3.3. General Simulation Design

The number of communities in this experiment sets to be 2 ($|C| = 2$). The number of actors in the first community c_1 sets to be 5881 ($NOA_{c_1} = 5881$). The number of transactions to perform sets to be 21532 ($NOT_{c_1} = 21532$). The number of actors in the second community c_2 sets to be 3162 ($NOA_{c_2} = 3162$). The number of transactions to perform sets to be 10139 ($NOT_{c_2} = 10139$). The cooperative reciprocal rating for c_1 sets to be 0.79 and for c_2 set to be 0.83 ($Crr_{c_1} = 0.79$ & $Crr_{c_2} = 0.83$). Also, the rate of untrustworthy actors in c_1 is 0.50 whereas in c_2 is 0.25 ($Uar_{c_1} = 0.50$ & $Uar_{c_2} = 0.25$). The malicious behavior rate for c_1 set to be 0.50 and for c_2 set to be 0.25 ($Mbr_{c_1} = 0.50$ & $Mbr_{c_2} = 0.25$). c_1 is chosen to have Sybil attack targeting a single actor, while c_2 is chosen to be with no Sybil attack.

7.3.4. Sybil Attack Design

A single untrustworthy actor that owns a single identity usually acts malicious at a certain time t targeting another actor. This type of attack is simple and straightforward with limited influence. Therefore, we designed a sophisticated attack consisting of a collection of identities aiming to mislead the community. If we look at the community as a graph with vertices and edges, the Sybil attack creates a bunch of edges with dishonest weights (deceitful evaluations). Sybil actors (*Sya*) provide fake feedback ($R = \{r_1, r_2, \dots, r_n\}$) in a given time frame ($T = \{t_1, t_2, \dots, t_n\}$) in order to improve (Support) or ruin the reputation of a target actor. The time frame for all transactions and evaluations falls between the range [1289380982, 1292361186] in an epoch time format. Sybil actors are uniformly picked random from untrustworthy actors. The Sybil actors rate sets to be 0.01 (1%) of the total number of untrustworthy actors ($Syr = 0.01$). Since $NOA_{c_1} = 5881$ actors with $Uar_{c_1} = 0.50$, the number of Sybil actors is 30 ($NSA_{c_1} = 30$). We additionally assume that the target actor (Ta) exists in the two communities ($Ta \ni c_1 \; \& \; Ta \ni c_2$) and as designed all *Ta's* evaluations are ended up at the trust DApp ($\forall r_{i(Ta)} \in R_{Ta}$).

7.3.5. Sybil Evaluation

We randomly have chosen three actors as targets of the Sybil attack. The actors' IDs were 150, 2871, and 2944. The number of attempts to carry out transactions with every target actor was 30 as every Sya tried to perform only one transaction at time t with the target actor. Based on the trust score of each Sya, the target accepts or rejects dealing with the corresponding party. The total number of performed transactions with the actor holding ID# 150 was13 transactions. The total number of performed transactions with the actor holding ID# 2871 was 17 transactions whereas the total number of carried out transactions with the actor holding ID# 2944 was 14. We assume that the same Sybil actors will target the three actors. The ratings given by Sybil actors to target actors for all transactions were negative and randomly generated from the range [-1, -10]. Sybil actors usually want to conceal their maliciousness to the community, so we generate negative ratings randomly to not create an easily detected malicious pattern. Table 28 shows a list of the chosen Sybil actors and table 29 shows the plan of the three Sybil attacks and the evaluations given by Sybil actors. Figures 81, 82, and 83 graphically show the chronological order of the Sybil attacks against actors Ids: 150, 2871, and 2944. Attack power (AP) indicates the damage attribute derived from the attack strength which increases over time until reaching 100%.

Table 28. Sybil Actors.

Sybil Actor	Sya_1	Sya_2	Sya_3	Sya_4	Sya_5	Sya_6	Sya_7	Sya_8	Sya_9	Sya_{10}	Sya_{11}	Sya_{12}
Id	4	11	33	94	144	210	457	603	689	758	970	991
Sybil Actor	Sya_{13}	Sya_{14}	Sya_{15}	Sya_{16}	Sya_{17}	Sya_{18}	Sya_{19}	Sya_{20}	Sya_{21}	Sya_{22}	Sya_{23}	Sya_{24}
Id	1215	1628	1748	1921	2198	2322	2637	3149	3680	3707	4002	4075
Sybil Actor	Sya_{25}	Sya_{26}	Sya_{27}	Sya_{28}	Sya_{29}	Sya_{30}						
Id	4177	4520	4647	5018	5685	5799						

Table 29. Details of the Three Principal Malicious Attacks (Sybil Attacks).

Ta.Id: 150				*Ta.Id:* 2871				*Ta.Id:* 2944			
Sya	*Sya.Id*	*r*	*t*	*Sya*	*Sya.Id*	*R*	*t*	*Sya*	*Sya.Id*	*r*	*t*
Sya_1	4	-8	1289555746	Sya_1	4	-5	1289441451	Sya_3	33	-1	1289441451
Sya_3	33	-10	1289556989	Sya_2	11	-1	1289489668	Sya_5	144	-5	1289555871
Sya_4	94	-1	1289660347	Sya_3	33	-7	1289555746	Sya_6	210	-4	1289641704
Sya_5	144	-6	1289770757	Sya_5	144	-8	1289556989	Sya_7	457	-9	1289660347
Sya_6	210	-2	1289873368	Sya_6	210	-3	1289641704	Sya_9	689	-9	1289873368
Sya_9	689	-9	1290197554	Sya_7	457	-7	1289770757	Sya_{13}	1215	-7	1290666953
Sya_{10}	758	-8	1290644458	Sya_9	689	-7	1290090118	Sya_{14}	1628	-5	1291157823
Sya_{12}	991	-3	1290666953	Sya_{10}	758	-4	1290197554	Sya_{15}	1748	-2	1291217882
Sya_{20}	3149	-7	1290758643	Sya_{12}	991	-2	1290644458	Sya_{16}	1921	-4	1291578872
Sya_{21}	3680	-6	1290826591	Sya_{18}	1628	-8	1290758643	Sya_{19}	2637	-6	1291684580
Sya_{25}	4177	-2	1291055973	Sya_{20}	3149	-10	1291578872	Sya_{22}	3707	-3	1291757016
Sya_{28}	5018	-4	1291578872	Sya_{21}	3680	-5	1291591212	Sya_{23}	4002	-1	1292099161
Sya_{29}	5685	-6	1292099161	Sya_{23}	4002	-1	1291757016	Sya_{25}	4177	-4	1292196407
-	-	-	-	Sya_{25}	4177	-9	1292071379	Sya_{29}	5685	-2	1292206272
-	-	-	-	Sya_{26}	4520	-6	1292193828	-	-	-	-
-	-	-	-	Sya_{28}	5018	-10	1292206272	-	-	-	-
-	-	-	-	Sya_{29}	5685	-4	1292361186	-	-	-	-

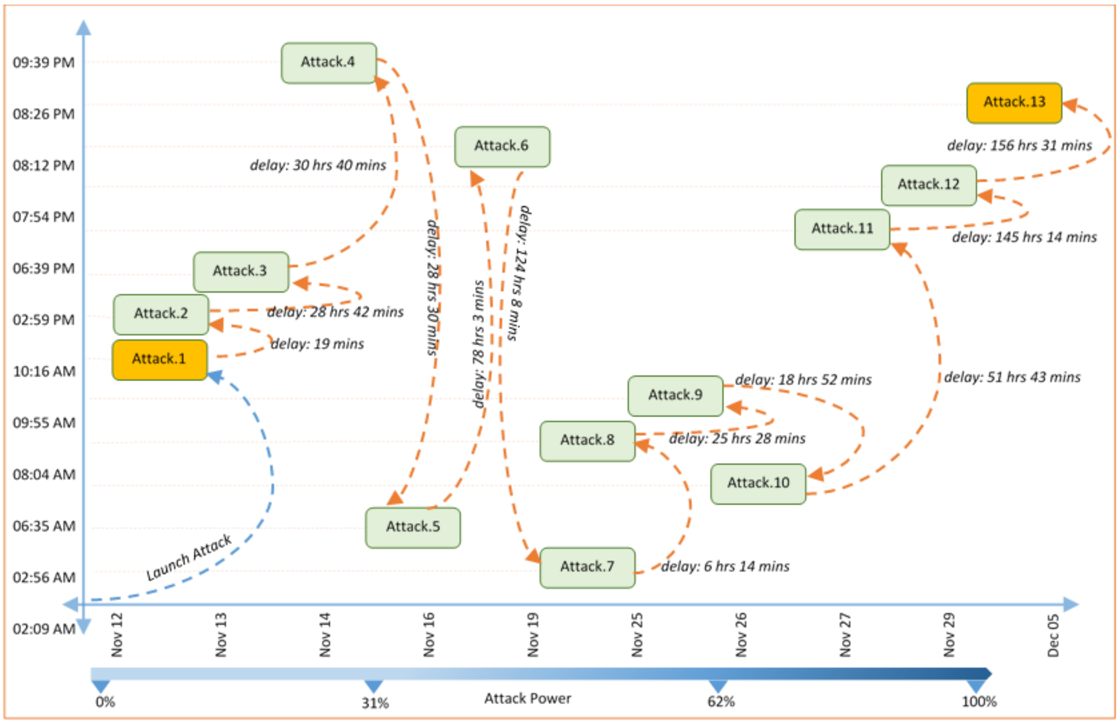


Figure 81. The Chronological Order of a Sybil Attack Against Actor# 150.

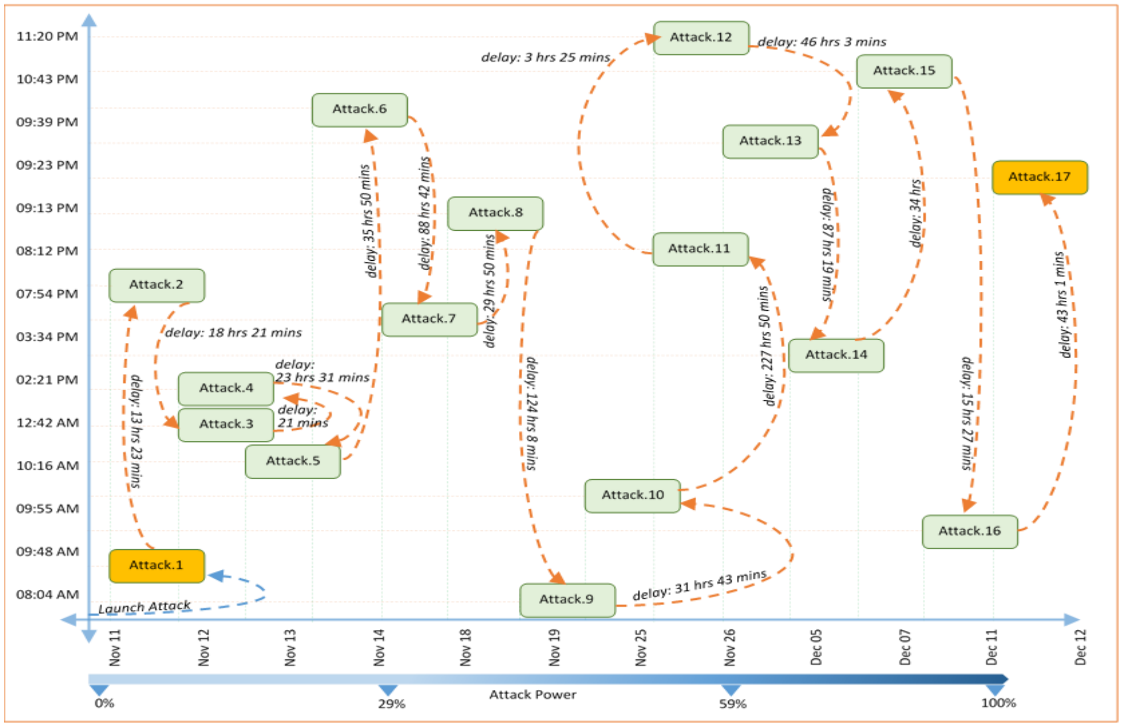


Figure 82. The Chronological Order of a Sybil Attack Against Actor# 2871.

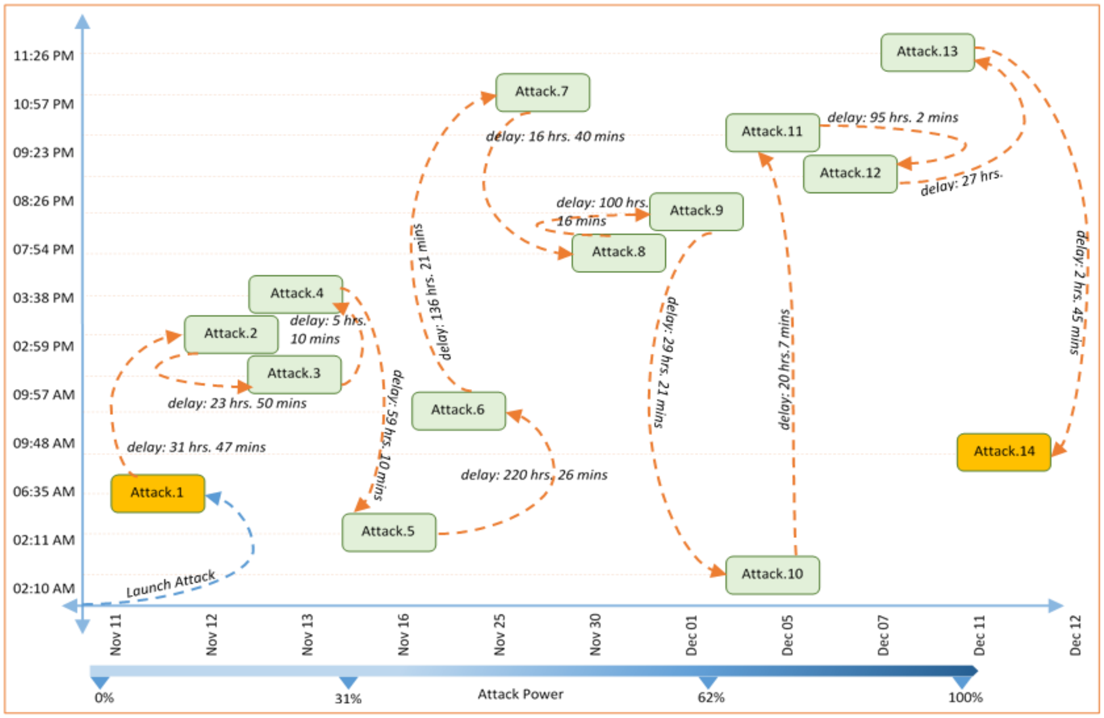


Figure 83. The Chronological Order of a Sybil Attack Against Actor# 2944.

- ***Simulation Results***

Figures 84-89 show the trust evolvement of actors holding numbers 150, 2871, and 2944. Those actors are targets of Sybil Attacks. This experiment varies from previous experiments. Previous experiments conduct non-collusive scenarios, whereas this experiment conducts collusive scenarios. We consider a Sybil attack as a type of collusive attacks. We present the trust evolvement in the case of using blockchain technology forming cross-community trust and in the case of a single-community trust without blockchain. The goal of this experiment is to show the effectiveness of using blockchain in alleviating the impact of malicious attacks compared to a model that does not employ such technology.

When launching the Sybil attack targeting actor holding Id 150, the results shown in figures 84 and 85 demonstrate that the trust score decreased gradually in the instance of single-community more than in the instance of cross-community using blockchain. This can be seen in table 30 as trust score averages of TrustMe are 0.82 when using the cross-community form and 0.65 when using the single community form. Trust score averages of PeerTrust are 0.72 for cross-community and 0.61 for single-community. Also, trust score averages of RawMean are 0.28 for cross-community and 0.23 for single-community. This indicates that the cross-community design is better than the single-community design. Another important observation is that the RawMean model is heavily affected in both cases with an advantage given to the cross-community design. This is another proof that RawMean is not effective in all cases, and it is highly vulnerable in collusive and non-collusive scenarios. PeerTrust and TrustMe models achieve superior performance when using cross-community.

There is an additional allusion that factors ascribed to a trust model help to lessen the impact of unusual behaviors. This conclusion can likewise be observed when monitoring the trust score averages of actors holding Ids 2871 and 2944. Further, the advantage of the cross-community design supplements is that it increases the size of knowledge which helps to detect malicious behaviors and to produce more precise trust scores. Indeed, the blockchain model abates the influence of Sybil attack and makes the attack ineffective and unsuccessful. The blockchain model strengthens the internal immune system of trust models.

Table 30. Averages of Trust Scores in Cross-Community and Single-Community Trust.

Actor Id	TrustMe		PeerTrust		RawMean	
	Cross-Community	*Single-Community*	*Cross-Community*	*Single-Community*	*Cross-Community*	*Single-Community*
150	0.82	0.65	0.72	0.61	0.28	0.23
2871	0.80	0.73	0.64	0.55	0.24	0.20
2944	0.88	0.76	0.80	0.64	0.39	0.34

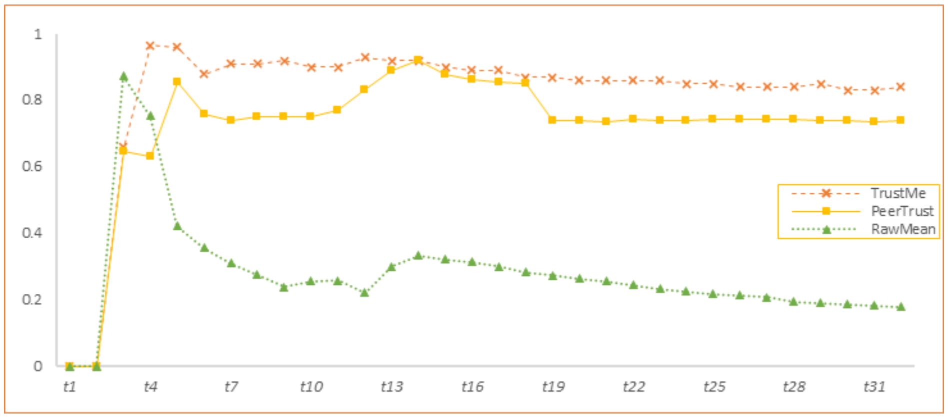


Figure 84. Trust of Actor# 150 - Exposure to Sybil Attack in Blockchain-Based Design.

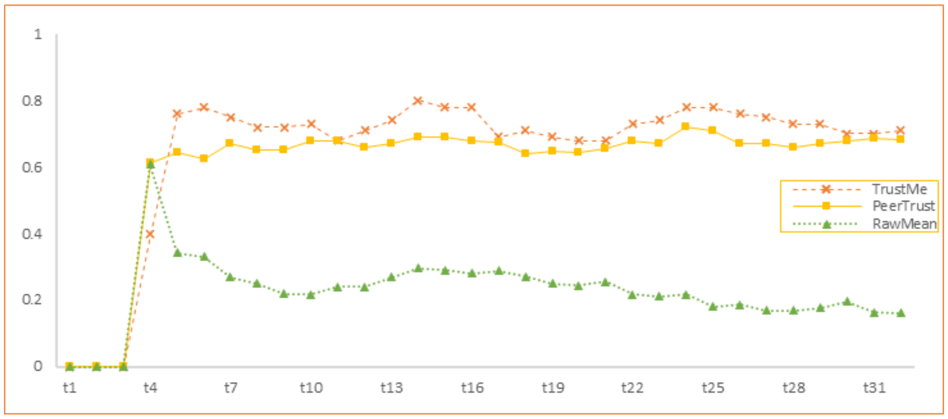


Figure 85. Trust of Actor# 150 - Exposure to Sybil Attack in Unshared-Reputation Design.

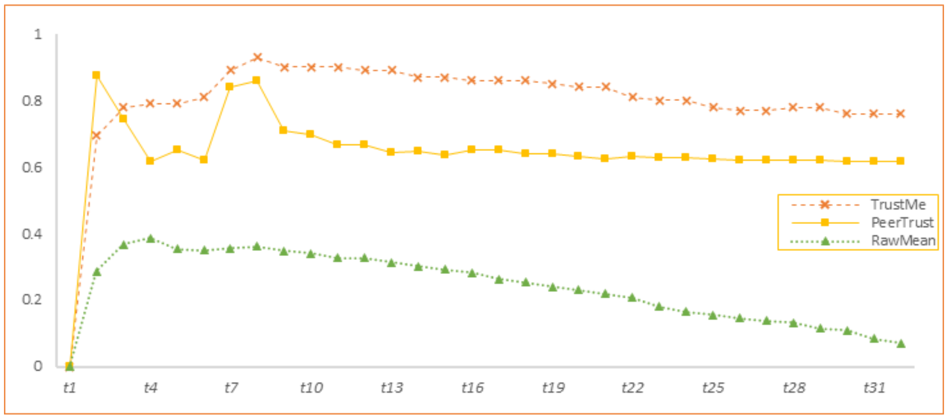


Figure 86. Trust of Actor# 2871 - Exposure to Sybil Attack in Blockchain-Based Design.

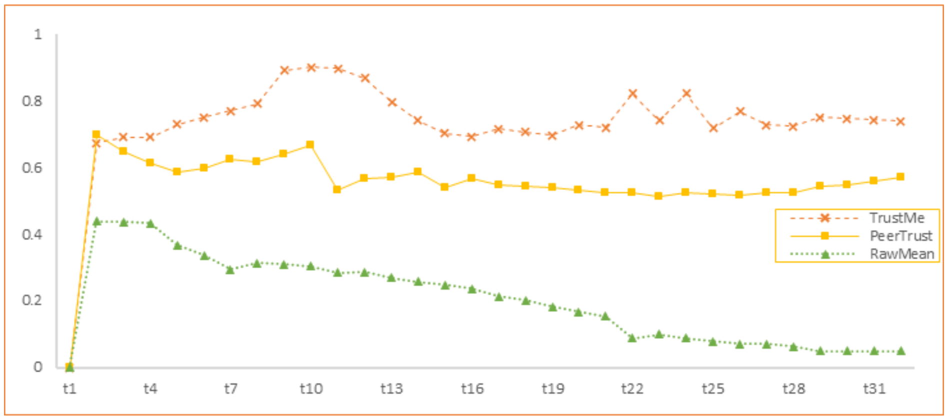


Figure 87. Trust of Actor# 2871 - Exposure to Sybil Attack in Unshared-Reputation Design.

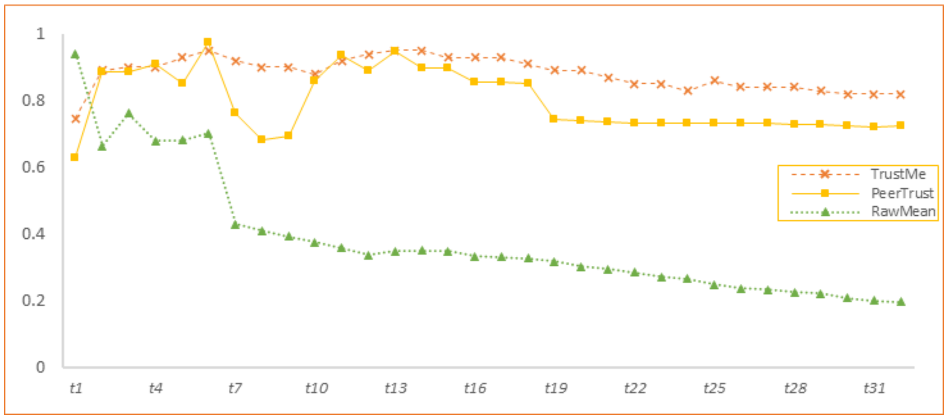


Figure 88. Trust of Actor# 2944 - Exposure to Sybil Attack in Blockchain-Based Design.

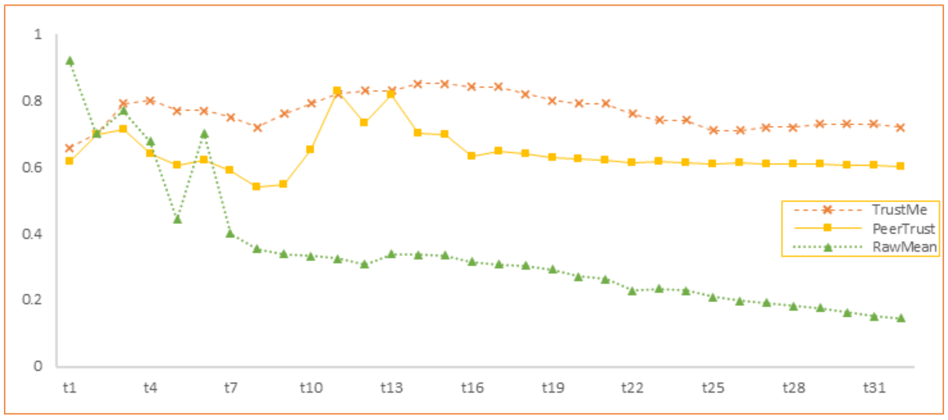


Figure 89. Trust of Actor# 2944 - Exposure to Sybil Attack in Unshared-Reputation Design.

8. CONCLUSIONS, FUTURE WORK, AND RECOMMENDATIONS

8.1. Conclusions

Virtual communities provide a unique climate where interactions performed among its members/actors who usually have similar interests. Actors can be clustered as trustworthy and untrustworthy. Because of the presence of untrustworthy actors, the community will likely be subjected to malicious attacks. The growth of malicious attacks increases uncertainty and risk in these communities. Trust and reputation are substantial elements that need to exist to reduce such uncertainty and risk. To ensure safe operation, it is extremely important to select a trustworthy actor to deal with. Ascertaining the trust of an actor is quite crucial in preventing counterfeit transactions, minimizing risk, and avoiding illicit actors. A Trust mechanism can serve as an observer to track the trust dynamism based on actors' behavior supplying precise trust information.

Also, trust as a private asset should be obsessed and managed by its legitimate actor. Due to the centralization of the most online communities, the ownership feature is not guaranteed, and the community manage and control these assets. This may lead to losing trust assets or be used improperly without permission. Sharing information is another issue that needs to be taken into account to take advantage of its availability. Communities can partner and share trust information to rapidly and facilely detect perilous behavior. Shared trust information further increases the preciseness of the generated trust scores. The issues of trust, ownership, and sharing trust information should be addressed to construct more secure, integrable, and reliable communities. The contribution of this study is to design and test new promising models that aim to mitigate these problems and to make virtual communities safer.

8.1.1. Study Contributions

- ***TrustMe Model: Reducing Impact of Malicious Attacks***

Trust is a significant factor in establishing and building a healthy reputation among actors. This thesis provided a reputation-based trust model that contributed to reducing the uncertainty levels among actors in a peer-to-peer environment. Malicious actors may tend to deceive to change the reputation of one peer. Without such a trust model, malicious actors can achieve their goals easily. This model mainly aims to generate trust scores for actors through scanning the history of the actors' transactions/interactions over time. The model can be applied by the actors themselves and online platforms as well. Before performing a transaction, participants can explore the trust score of each other by using this model as a tool, and this is to ease the process of decision-making. Actors, in this context, can lessen the likelihood of being victims of fraud.

Furthermore, online communities are vulnerable to scams and frauds which lead to the necessity of trust models. Actors are highly subjected to malicious attacks or dealing with dishonest untrustworthy actors. Thus, this thesis developed a model that was designed with several factors focusing on reducing such malicious attacks. To do that, the impact of outlier ratings was minimized in the model. We found that these outliers are quite often a product of launching a malicious campaign in conjunction with temporal information to upgrade or degrade the reputation of the target actor.

In this regard, a collection of experimental analytic was conducted starting from density-based clustering, graph analysis, to building a set of simulations. Density-based clustering as a pilot study shows that most of the trust scores were correctly clustered measured by the Silhouette coefficient and the sum of squared errors. Another study was conducted using graph theory to analyze the produce trust scores by TrustMe with the values of the network metrics. The results show that TrustMe sounds to be more logical in producing trust scores that are consistent and correlated with the values of network measures. For instance, if the score of betweenness is high, the trust score is high. The same case is applied to the eigenvector. A set of four simulation experiments was also conducted to measure the evaluation accuracy and the transaction success rate of the trust models where three models were used for comparison purposes. The simulation was controlled by some variables such as the population of the community, the number of transactions, the malicious behavior rate, and the number of the untrustworthy actor, and this to emulate different scenarios. The results find that TrustMe outperforms RawMean and PeerTrust model in all settings by acquiring the highest values of trust evaluation accuracy (*TEA*) and transaction success rate (*TSR*). The hypothesis regarding the problem of biased/malicious reviews stated that the TrustMe model mitigates the problem of biased/malicious reviews. The results illustrate that there was a statistically significant difference between the hypothetical *TEA* and *TSR* and the observed *TEA* and *TSR*. Therefore, we reject the null hypothesis and accept the alternative hypothesis that confirms that TrustMe helps in lessening biased/malicious reviews.

- ***Blockchain-Based Trust Model: Cross-Community Trust***

Another important contribution is leveraging information sharing across communities to allow actors to make their trust assets mobile and to be used in multiple virtual communities. Sharing information further increases the size of knowledge which in turn enlarges the accuracy of the produced trust scores. This assists in minifying malicious behavior and discloses untrustworthy actors. This also accelerates securing multiple communities once as the information of detected untrustworthy actors will be passed to all partnered communities. Here, blockchain becomes prominent as a possible solution to such issues by connecting multiple separate online platforms to a decentralized platform that preserves all trust information. This station can provide trust information to anyone who inquiries about a certain actor. Indeed, blockchain technology offers some important values: traceability, transparency, and immutability. Traceability means all trust information can be traced back to see the history of an actor's trust (trust evolvement). Whereas, transparency implies that trust scores are available, reachable, and visible to everyone. Immutability indicates that the stored trust information became tamper-proof. One possible scenario to show the benefit of trust information sharing is that there is an actor who already has an account on the first platform while no account on the second one. When the actor decided to open a new account on the second platform that partnered with the first platform, the second platform can learn about the trust information of this actor without previous local transactions. This can be done by querying the blockchain-based trust model about the trust information of that actor. Hence, such a decentralization mechanism helps actors to move their trust information from one community to another without much effort. Participants and the operators of the platforms, in this context, can track actors' behavior which allows them quickly disclosing untrustworthy actors who aim to provide destructive information to mislead the community.

In this regard, an experimental analysis was conducted to discover the effectiveness of using the blockchain-based trust model in terms of a cross-community trust. A simulation system was built consisting of a set of stochastic models including an adversarial model mimicking three Sybil attacks and cross-community trust information. This simulation produced trust scores for actors using three trust models including the *TrustMe* model. Three actors were chosen randomly to be targets of the designed Sybil attack. The results show that TrustMe outperformed other models because the impact of the malicious attack was marginal.

The hypothesis regarding the problem of the efficiency of managing multiple trusts and the impact of malicious cyber-attack stated that the blockchain-based trust model efficiently manages multiple trusts and alleviates the impact of malicious attacks. The results demonstrate that there was a statistically significant difference between the hypothetical *TEA* and *TSR* and the observed *TEA* and *TSR.* Therefore, we reject the null hypothesis and accept the alternative hypothesis that confirms that the blockchain-based trust model aids in managing multiple trusts and alleviating the impact of malicious attacks.

- ***Blockchain-Based Trust Model: Ownership***

This study further provides a model that focuses on managing identity from the angle of trust and reputation since there is no enough work highlighting the importance of cross-community trust management using blockchain. Some identity management principles have a strong connection with managing multiple trust and reputations. Therefore, reputation management can get some benefits from applying some identity management strategies and principles. In some scenarios, it is unfair to have multiple identities and multiple reputations for the same actor, particularly in e-commerce communities. For instance, if one seller wants to control his/her trust information by conveying it form a community to another, this actor cannot perform such action because of losing control. This control is given to a central authority where trust information is stored and managed. An additional scenario is that actors may lose a long history of performing transactions with behaving honestly. They might have built a strong reputation over time, the host community decided for whatever reason to stop running their business. In this case, actors do not own their trust assets, and they will lose their long reputable record in a short time.

This study provides a design of a blockchain-based trust model in the form of a distributed application called *"distributed application of cross-community trust (DACCT)"* to store and manage trust information in decentralization fashion. This design takes into consideration the ownership issue to be returned to the legitimate owner. Here, actors can reach their trust information any time from any location using DACCT. Also, others as individuals or communities can access to trust information by providing the address of the actor under questioning. This design guarantees that the ownership feature is delivered to its authentic actor. Other principles have been taken into account in this design and classified as indispensable elements such as privacy. One of the public blockchain characteristics is transparency which implies that all information is visible and accessible. Thus, to protect private and sensitive information, a piece of DACCT design concerns this issue by using a Proxy design pattern. This design allows only permissioned actors to access sensitive information.

To sum up, this study provides a long run experiment proving that the TrustMe model alone is productive and efficient in generating trust scores. All experiments provide strong evidence about the effectiveness of TrustMe. If the TrustMe model is integrated with blockchain mechanisms, its performance will be more effective in reducing the influence of malicious attacks in virtual communities. Also, by using blockchain technology in applying cross-community trust, trust information will be more precise and available to anyone, it cannot be lost and managed by a decentral mechanism which grants ownership to its possessor through DACCT. These solutions make online communities more secure, trusted, and reliable against misleading information and malicious behavior.

8.2. Future Work and Recommendations

In this study, the TrustMe model was developed as an efficient trust tool to protect virtual communities and help actors making a proper decision based on the given trust information. Trust is a complicated concept that needs to be investigated further in different forms, such as addressing the cold-start problem by proposing models can deal with different situations of cold-start issue. Another potential future work that is somehow connected to the cold-start issue is group-based stereotypes to trust unknown members. This kind of work can be conducted using graph analysis by extracting sub-communities from the main community to study the newly joined actors, and what are the shared characteristic that made this actor to clustered in such community. Here, the new actor can get a trust score based on common characteristics.

Moreover, if we only consider building a complete distributed application (DApp) using Ethereum blockchain, there is a crucial desire to establish a new consensus mechanism to decentralize community management completely. This consensus concerns monitoring the behavior of communities in the case of providing misleading information and concerns communities' membership processes. This work will introduce an effective mechanism that can be helpful for cross-community-oriented applications.

REFERENCES

[1] J. E. Youll, "Peer to peer transactions in agent-mediated electronic commerce," *Master's thesis, Massachusetts Institute of Technology,* 2001.

[2] S. Mohammadi, H. Mirvaziri, M. Ghazizadeh-Ahsaee, and H. Karimipour, "Cyber intrusion detection by combined feature selection algorithm," *Journal of Information Security and Applications,* vol. 44, pp. 80-88, 2019.

[3] J. T. Hancock and J. Guillory, "Deception with technology," in *The Handbook of the Psychology of Communication Technology*, John Wiley & Sons, Inc, 2015, p. 270–289.

[4] B. Friedman, P. H. Khan, and D. C. Howe, "Trust online," *Communications of the ACM,* vol. 43, no. 12, pp. 34-40, 2000.

[5] B. Yu and M. P. Singh, "A social mechanism of reputation management in electronic communitie," *Cooperative Information Agents IV (CIA) - The Future of Information Agents in Cyberspace, Springer* , vol. 1860, pp. 154-165, 2000.

[6] J. Sabater and C. Sierra, "Regret: A reputation model for gregarious societies," in *In Proceedings of the fifth international conference on Autonomous agents (AGENTS '01). Association for Computing Machinery*, New York, NY, USA, 2001.

[7] Li Xiong and Ling Liu, "A reputation-based trust model for peer-to-peer ecommerce communities," in *In Proceedings of the 4th ACM conference on Electronic commerce (EC '03)*, Newport Beach, CA, USA, 2003.

[8] T. D. Huynh, N. R. Jennings, and N. R. Shadbolt, "An integrated trust and reputation model for open multi-agent systems," *Autonomous Agents and Multi-Agent Systems,* vol. 13, no. 2, p. 119–154, 2006.

[9] A. Salehi-Abari and T. White, "Towards con-resistant trust models for distributed agent systems," in *Proceedings of the Twenty-first International Joint Conference on Artificial Intelligence (IJCAI '09)*, 272–277.

[10] A. Salehi-Abari and T. White, "Trust models and con-man agents: From mathematical to empirical analysis," in *Proceedings of the Twenty-Fourth AAAI Conference on Artificial IntelligenceJuly (AAAI'10),* Atlanta, Georgia, USA, 2010.

[11] J. Cho, K. Chan, and S. Adali, "A survey on trust modeling," *ACM Computing Surveys,* vol. 48, no. 2, pp. 28.1-28.40, 2015.

[12] J. Bobadilla, F. Ortega, A. Hernando, and J. Bernal, "A collaborative filtering approach to mitigate the new user cold start problem," *Knowledge-Based Systems,* vol. 26, p. 225–238, 2012.

[13] D. Cosley, S. K. Lam, I. Albert, J. A. Konstan, and J. Riedl, "Is seeing believing?: How recommender system interfaces affect users' opinions," *ACM CHI '03,* p. 585–592, 2003.

[14] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock, "Methods and metrics for cold-start recommendations," in *In Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '02)*, New York, NY, 2002.

[15] P. Dunphy and F. A. Petitcolas, "A first look at identity management schemes on the blockchain," *IEEE Security & Privacy*, vol. 16, no. 4, pp. 20-29, 2018.

[16] S. Karunakaran, K. Thomas, E. Bursztein, and O. Comanescu, "Data breaches: User comprehension, expectations, and concerns with handling exposed data," *USENIX Symposium on Usable Privacy and Security (SOUPS),* no. ., pp. 217-234, 2018.

[17] H. S. Jones and W. Moncur, "The role of psychology in understanding online trust," in *Psychological and Behavioral Examinations in Cyber Security*, IGI Global, 2018, pp. 109-132.

[18] B. Qureshi, G. Min, and D. Kouvatsos, "Collusion detection and prevention with FIRE trust and reputation model," in *10th IEEE International Conference on Computer and Information Technology*, Bradford, UK, 2010.

[19] "ACI Worldwide," 17 January 2017. [Online]. Available: https://www.aciworldwide.com/news-and-events/press-releases/2017/january/global-fraud-attempts-increased-by-31-during-holiday-shopping-season. [Accessed 25 03 2019].

[20] "Trends in e-commerce & digital fraud: Mitigating the risks," Radial & Eknresearch, Deerfield, IL, 2017.

[21] S. Bok, Lying: Moral choice in public and private Life, updated edition, New York: Vintage Books, 1999.

[22] D. Gambetta, "Can we trust trust?," *Trust: Making and Breaking Cooperative Relations, Department of Sociology, University of Oxford*, pp. 213-237, 2000.

[23] A. Abdul-Rahman and S. Hailes, "Supporting trust in virtual communities," in *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, Maui, HI, USA, 2000.

[24] M. Fasli, Agent technology for ecommerce, Hoboken, NJ, USA: John Wiley & Sons, Inc., 2007.

[25] A. A. Brandão, L. Vercouter, S. Casare, and J. Sichman, "Exchanging reputation values among heterogeneous agent reputation models," in *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems (AAMAS 07),* 2007.

[26] R. K. Dash, N. R. Jennings, and D. C. Parkes, "Computational mechanism design: a call to arms," *IEEE Intelligent Systems,* vol. 18, no. 6, p. 40–47, 2003.

[27] S. D. Ramchurn, T. D. Huynh, and N. R. Jennings, "Trust in multi-agent systems," *The Knowledge Engineering Review,* vol. 19, pp. 1-25, 2004.

[28] W. Jansen and T. Karygiannis, "Mobile agent security," *NIST,* pp. 800-19, 1999.

[29] C. Castelfranchi and R. Falcone, "Principles of trust for MAS: cognitive anatomy, social importance, and quantification," in *Proceedings International Conference on Multi Agent Systems (Cat. No.98EX160),* Paris, France, 1998.

[30] P. Dillenbourg, "Distributing cognitive over humans and machines," *International Perspectives on the Psychological Foundations of Technology- Based Learning Environments,* pp. 165-184, 1996.

[31] T. Ishaya and D. P. Mundy, "Trust development and management in virtual communities," in *Trust Management (iTrust 2004), Springer*, Berlin, Heidelberg, 2004.

[32] "Amazon, Review System," [Online]. Available: http://www.amazon.com. [Accessed 26 March 2017].

[33] "eBay, Review System," [Online]. Available: http://www.ebay.com.. [Accessed 26 March 2017].

[34] J. Sabater and C. Sierra, "Social ReGreT, a reputation model based on social relations," *ACM SIGecom Exchanges,* vol. 3, p. 44–56, 2001.

[35] D. Carboni, "Feedback based reputation on top of the bitcoin blockchain," *Cornell University Library,* 2015.

[36] P. Resnick, K. Kuwabara, R. Zeckhauser, and E. Friedman, "Reputation systems," *Communications of the ACM*, vol. 43, no. 12, p. 45–48, 2000.

[37] Y. Wang and J. Vassileva, "Trust and reputation model in peer-to-peer networks," in *Proceedings Third International Conference on Peer-to-Peer Computing (P2P2003),* Linkoping, Sweden, 2003.

[38] D. L. Sackett, "Bias in analytic research," *Journal of Chronic Diseases*, vol. 32, no. 1-2, p. 51–63, 1979.

[39] I. Brocas and J. D. Carrillo, "Dual-process theories of decision-making: a selective survey," *ELSEVIER - Journal of Economic Psychology,* vol. 41, pp. 45-54, 2014.

[40] S. Watts, "Application of dual-process theory to information systems: Current and future research directions," *Foundations and Trends in Information Systems,* vol. 1, no. 2, pp. 69-162, 2015.

[41] S. Kloker, S. Bluhm, and M. Regener, "Application of the dual-process theory to debias forecasts in prediction markets," 2017.

[42] S. Chaiken, "Heuristic versus systematic information processing and the use of source versus message cues in persuasion," *APA PsycArticles,* vol. 39, no. 5, p. 752–766, 1980.

[43] F. Marquart and B. Naderer , "Communication and persuasion: central and peripheral routes to attitude change," *Springer VS,* pp. 231-242, 2016.

[44] P. J. Kitchen, G. Kerr, D. E. Schultz, R. S. McColl, H. Pals, "The elaboration likelihood model: review, critique and research agenda," *European Journal of Marketing,* vol. 48, p. 2033–2050, 2014.

[45] J. Nel and C. Boshoff, "Development of application-based mobile-service trust and online trust transfer: an elaboration likelihood model perspective," *Behaviour and Information Technology,* vol. 36, no. 8, p. 809–826, 2017.

[46] S. Chen , K. Duckworth, and S. Chaiken, "Motivated heuristic and systematic processing," *Psychological Inquiry,* vol. 10, no. 1, pp. 44-49, 1999.

[47] R. Sikora and L. You, "Effect of reputation mechanisms and ratings biases on traders ' behavior in online marketplaces," *Organizational Computing and Electronic Commerce,* vol. 24, no. 1, pp. 58-73, 2014.

[48] G. Kapoor and S. Piramuthu, "Sequential bias in online product reviews," *Journal of Organizational Computing and Electronic Commerce,* vol. 19, pp. 85 - 95, 2009.

[49] S. Piramuthu, G. Kapoor, W. Zhou, and S. Mauw, "Input online review data and related bias in recommender systems," *Decision Support Systems*, vol. 53, no. 3, p. 418–424, 2012.

[50] R. T. Sikora, and K. Chauhan, "Estimating sequential bias in online reviews : A Kalman filtering approach," *Elsevier - Knowledge-Based Systems*, vol. 27, p. 314–321, 2012.

[51] Z. Trifa and M. Khemakhem, "Sybil nodes as a mitigation strategy against Sybil attack," *Procedia Computer Science,* vol. 32, p. 1135–1140, 2014.

[52] J. R. Douceur, "The Sybil attack," *In Proceedings for the 1st International Workshop on Peer-to-Peer Systems (IPTPS), Springer,* vol. 2429, pp. 251-260, 2002.

[53] L. Detweiler, "The snakes of medusa – internet identity subversion," *Cypherpunks mailing lists,* 1993.

[54] G. Danezis and S. Schiffner, "Sybil attacks and reputation systems," *DIMACS Workshop on Information Security Economics*, pp. 18-19, 2006.

[55] K. Hoffman, D. Zage, and C. Nita-Rotaru, "A survey of attack and defense techniques for reputation systems," *ACM Computing Surveys*, vol. 42, no. 1, 2009.

[56] K. Gunderson, "Robots, consciousness, and programmed behaviour," *The British Journal for the Philosophy of Science*, vol. 19, no. 2, p. 109–122, 1968.

[57] E. Carrara, G. Hogben, and ENISA, "Reputation-based Systems: a security analysis," ENISA Position Paper No.2, 2007.

[58] D. A. Norman, "Psychology of everyday action," in *The Design of Everyday Things,* New York, Basic Books, 1988, pp. 34-53.

[59] M. Risius and K. Spohrer, "A blockchain research framework," *Business & Information Systems Engineering,* vol. 59, p. pages385–409, 2017.

[60] S. Nakamoto, "'Bitcoin: A peer-to-peer electronic cash system," *Tech. Rep.,* 2008.

[61] F. Glaser, "Pervasive decentralisation of digital infrastructures: A framework for blockchain enabled system and use case analysis," in *Proceedings of the 50th Hawaii International Conference on System Sciences (HICSS),* 2017.

[62] K. Sultan , U. Ruhi and R. Lakhani, "Conceptualizing blockchains: Characteristics & applications," *11th IADIS International Conference Information Systems,* vol. abs/1806.03693, 2018.

[63] A. M. Antonopoulos, Mastering bitcoin: Unlocking digital crypto-currencies, 1 edition, O'Reilly Media, Inc., 2014.

[64] M. A. Khan and K. Salah, "IoT security: Review, blockchain solutions, and open challenges," *Future Generation Computer Systems, Elsevier,* vol. 82, p. 395–411, 2018.

[65] "How does bitcoin work?," The-Bitcoin-Foundation, [Online]. Available: https://bitcoin.org/en/how-it-works. [Accessed 23 June 2019].

[66] "Block - Bitcoin Wiki," BitInfoCharts, [Online]. Available: https://en.bitcoin.it/wiki/Block. [Accessed 23 June 2019].

[67] "Ethereum average blocktime chart," EtherScan, [Online]. Available: https://etherscan.io/chart/blocktime. [Accessed 27 June 2019].

[68] F. Casino, T. K. Dasaklis, and C. Patsakis, "A systematic literature review of blockchain-based applications: Current status, classification and open issues," *Telematics and Informatics,* vol. 36, p. 55–81, 2019.

[69] Z. Zheng, S. Xie, H. Dai, X Chen, H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," in *IEEE International Congress on Big Data (BigData Congress),* Honolulu, HI, 2017.

[70] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," *SRI International ACM Transactions on Programming Languages and Systems,* vol. 4, no. 3, pp. 382-401, 1982.

[71] J. J. Bambara and P. R. Allen, Blockchain a practical guide to developing business, law, and technology solutions, New York: McGraw-Hill Education, 2018.

[72] K. Driscoll, B. Hall, M. Paulitsch, P. Zumsteg, and H. Sivencrona, "The real Byzantine generals," in *The 23rd Digital Avionics Systems Conference (IEEE Cat. No.04CH37576),* Salt Lake City, UT, USA, 2004.

[73] U. W. Chohan, "Cryptocurrencies: A brief thematic review," *SSRN Electronic Journal,* 2017.

[74] C. Lustig and B. Nardi, "Algorithmic authority: The case of bitcoin," in *HICSS '15: 48th Hawaii International Conference on System Sciences*, Hawaii, 2015.

[75] C. Sas and I. E. Khairuddin, "Design for trust an exploration of the challenges and opportunities of Bitcoin users," *In: Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems,* p. 6499–6510, 2017.

[76] M. Fröwis and R. Böhme, "In code we trust?," in *Springer International Publishing - Data Privacy Management, Cryptocurrencies and Blockchain Technology*, 2017.

[77] A. Tobin and D. Reed, "The inevitable rise of self-sovereign identity," The Sovrin Foundation, 2016.

[78] M. Ali, J. Nelson, R. Shea, and M. J. Freedman, "Blockstack: A global naming and storage system secured by blockchains," in *USENIX Annual Technical Conference (USENIX ATC '16)*, Denver, CO, 2016.

[79] M. Alizadeh, S. Abolfazli, M. Zamani, S. Baharun, and K. Sakurai, "Authentication in mobile cloud computing: A survey," *Journal of Network and Computer Applications,* vol. 61, pp. 59-80, 2016.

[80] S. Y. Lim, M. L. Kiah, T. F. Ang, "Security issues and future challenges of cloud service authentication," *Acta Polytechnica Hungarica,* vol. 14, no. 2, pp. 69-89, 2017.

[81] P. Madsen, "Liberty ID-WSF People Service - federated social identity," Liberty Alliance Project, 2005.

[82] D. W. Chadwick, "Federated identity management," *International School on Foundations of Security Analysis and Design (FOSAD),* p. 96–120, 2009.

[83] B. Zwattendorfer, D. Slamanig, K. Stranacher, F. Hörandner, "A federated cloud identity broker-model for enhanced privacy via proxy re-encryption," in *Communications and Multimedia Security (CMS 2014), Springer*, Berlin, Heidelberg, 2014.

[84] E. Ghazizadeh, M. Zamani, J. Ab Manan, A. Pashang, "A survey on security issues of federated identity in the cloud computing," in *IEEE 4th International Conference in Cloud Computing Technology and Science (CloudCom)*, Taipei, Taiwan, 2012.

[85] R. Dennis and G. Owen, "Rep on the block: A next generation reputation system based on the blockchain," in *Proc. IEEE 10th International Conference for Internet Technology and Secured Transactions (ICITST)*, London, UK, 2015.

[86] A. Schaub, R. Bazin, O. Hasan, and L. Brunie, "A trustless privacy-preserving reputation system," in *Proc. ICT Systems Security and Privacy Protection. SEC 2016*, 2016.

[87] D. Calvaresi, V. Mattioli, A. Dubovitskaya, A. F. Dragoni, and M. I. Schumacher, "Reputation management in multi-agent systems using permissioned blockchain technology," in *Proc. IEEE/WIC/ACM Int. Conf. Web Intell. (WI),* 2018.

[88] N. Alexopoulos, S. M. Habib, and M. Mühlhäuser, "Towards secure distributed trust management on a global scale: An analytical approach for applying distributed ledgers for authorization in the IoT," in *Proc. IEEE/WIC/ACM Int. Conf. Web Intell. (WI),* 2018.

[89] I. Pinyol, J. Sabater-Mir, and G. Cun, "How to talk about reputation using a common ontology: From definition to implementation," in *Proc. of the Ninth Workshop on Trust in Agent Societies*, Hawaii, 2007.

[90] T. Grinshpoun, N. Gal-Oz, A.Meisels, and E. Gudes, "CCR: A model for sharing reputation knowledge across virtual communities," in *IEEE/WIC/ACM International*

Joint Conference on Web Intelligence and Intelligent Agent Technology, Milan, Italy, 2009.

[91] S. Kumar, F. Spezzano, V.S. Subrahmanian, and C. Faloutsos, "Edge weight prediction in weighted signed networks," in *IEEE 16th International Conference on Data Mining (ICDM)*, Barcelona, 2016.

[92] S. Kumar, B. Hooi, D. Makhija, M. Kumar, V.S. Subrahmanian, and C. Faloutsos., "REV2: Fraudulent user prediction in rating platforms," in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining (WSDM '18)*, New York, NY, USA, 2018.

[93] W. Zheng, L. Jin, "Online reputation systems in web 2. 0 era," in *AMCIS 2009. Lecture Notes in Business Information Processing, vol 36. Springer*, Berlin, Heidelberg, 2009.

[94] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Computational and Applied Mathematics*, vol. 20, p. 53–65, 1987.

[95] B. Zhu, Y. Xia, "Link prediction in weighted networks: A weighted mutual information model," *Plos one*, vol. 11, no. 2, pp. 1-13, 2016.

[96] Q. Jiang, C. Hu, L. Xu, "Fraud detection in B2B platforms using data mining techniques," in *Advanced Data Mining and Applications (ADMA 2012), Springer*, Berlin, Heidelberg, 2012.

[97] D. Hansen, B. Shneiderman, and M. Smith, Analyzing social media networks with NodeXL: Insights from a connected world, Morgan Kaufmann, 2011.

[98] S. P. Borgatti, M. G. Everett, and J. C. Johnson, Analyzing social networks, 1 edition, SAGE Publications Limited, 2013.

[99] U. Brandes, "A faster algorithm for betweenness centrality," *The Journal of Mathematical Sociology,* vol. 2, no. 25, p. 163–177, 2001.

[100] S. L. Zabell, "The rule of succession," *Erkenntnis,* vol. 31, no. 2-3, p. 283–321, 1989.

[101] L. C. Freeman, "A set of measures of centrality based on betweenness," *American Sociological Association,* vol. 40, no. 1, p. 35–41, 1977.

[102] P. Bonacich, "Power and centrality: a family of measures," *American Journal of Sociology,* vol. 92, no. 5, p. 1170–1182, 1987.

[103] S. Gago, J. C. Hurajová, and T. Madaras, "On betweenness-uniform graphs," *Czechoslovak Mathematical Journal,* vol. 63, no. 3, p. 629–642, 2013.

[104] B. Ruhnau, "Eigenvector centrality: A node centrality?," *Social Networks,* vol. 22, no. 4, pp. 357-365, 2000.

[105] P. Fouliras, "A novel reputation-based model for e-commerce," *Operational Research,* pp. 113-138, 2013.

[106] D. Z. Morris, "Bitcoin is not just digital currency. It's Napster for finance," Fortune, 21 January 2014. [Online]. Available: https://fortune.com/2014/01/21/bitcoin-is-not-just-digital-currency-its-napster-for-finance/. [Accessed 7 November 2018].

[107] R. Schulpen, "Smart contracts in the Netherlands," University of Tilburg, August 2018. [Online]. Available: http://arno.uvt.nl/show.cgi?fid=146860. [Accessed 7 October 2019].

[108] D. Tapscott and A. Tapscott, Blockchain revolution: How the technology behind bitcoin is changing money, business, and the world, vol. 7, Portfolio; Reprint edition, 2018, p. 275–276.

[109] M. Alharby, A. Aldweesh, and A. v. Moorsel, "Blockchain-based smart contracts: A systematic mapping study of academic research," in *International Conference on Cloud Computing, Big Data and Blockchain (ICCBB)*, Fuzhou, China, 2018.

APPENDIX A. EXTERNAL TRUST MODELS USED IN THE STUDY

- **RawMean Model**

It is mainly supervised and managed by a central authority. This reputation system is based on feedbacks that are given via a number of people. In e-commerce communities, there is significant use of feedback systems that are used to evaluate the rating of sellers/buyers or peers in a general environment such as a peer-to-peer system. Amazon, eBay, Netflix, and other online platforms, for example, apply feedback systems to allow people to evaluate each other expressing their satisfaction with a transaction in which they were involved. Many virtual communities employ a simple metric to compute rating scores over six or twelve months, and this metric can be vulnerable to malicious feedback. The following equation is used in eBay to generate a global single reputation score [34, 106]:

$$S = \frac{P}{(P+N)} * 100$$

Where *S* represents the satisfaction value of a seller or a buyer; *P* denotes the number of positive feedback; *N* denotes the number of negative feedback, and ignore the number of neutral feedback. In more detail, eBay uses a star system for the evaluation process in which people can rate each other using values from 0 to 5. In the case of giving 4 or 5 stars, eBay translates that to one positive score (+1), and in the case of giving three stars, it is converted to zero (0) as a neutral value, and it will not be used in the evaluation process. In the case of giving 1 or 2 stars, the system will count it as one negative feedback (-1).

- **PeerTrust Model**

Xiong and Liu [7] developed the PeerTrust model and stated that five factors should be used to compute a trust value. They have developed a reputation-based trust model that deals with the following factors: feedback, feedback scope, credibility transaction context factor, and community context factor. Their model is a general metric that can be adapted in different situations. In this model, the first factor is feedback, which is a major one that represents the review given by a buyer or seller after each transaction. The second factor is the feedback scope, which represents the total number of transactions. This factor has a vital impact on the trust score because by raising the volume of transactions, people might be able to conceal some of their malicious behaviors. Another factor is credibility, which, when used, can improve the accuracy of the trust score since this factor should reveal whether one person provides a false statement about others or not. Figuring out the credibility can limit the possibility of being vulnerable to dishonest feedbacks, which could cause a trustworthy person/peer to become untrustworthy because of a large number of dishonest feedback or reviews. The transaction context factor is another variable that should be taken into consideration. Xiong and Liu argue that transactions could have attributes values that distinguish one transaction from the other in the same e-commerce community. For instance, a seller might be honest with transactions that cost a small amount of money to receive a high rating score while being dishonest with transactions that cost a large amount of money. The last factor in this model is the community context factor, which makes the trust model adapted to different communities and situations. One example of the community context factor is offering incentives to raters, which means that when a rater delivers feedback, the rater receives an increase in his/her trust score.

- ***The general metric of PeerTrust***

$$T(u) = \alpha * \frac{\sum_{i=1}^{I(u)} S(u,i) * \ Cr\big(p(u,i)\big) * TF(u,i)}{I(u)} + \beta * CF(u)$$

where *T(u)* indicates the trust score of peer *u* at time *t*; *I(u)* denotes the total number of transactions; and *S(u, i)* represents the normalized amount of satisfactions peer *u* receives from *p(u, i)* in its i_{th} transaction. *Cr(u, i)* represents the credibility of the feedback submitted by *p(u, i)*, while *TF(u, i)* denotes the adaptive transaction context factor for peer u's i_{th} transaction. *CF(u)* denotes the adaptive community context factor of peer u during time t. α *and* β indicate the normalized weight factors for the two parts. The first part of the metric represents the average amount of credible satisfaction a peer receives for each transaction. The second part alters the first part by increasing or decreasing the trust score based on community-specific characteristics and situations. The basic metric consists of three factors (*S(u, i), Cr(u, i), and I(u)*) that should not be ignored in whatever case. To adapt the metric to be basic metric is by making $\alpha = 1$, $\beta = 0$, and *TF(u, i)* = 1:

$$T(u) = \frac{\sum_{i=1}^{I(u)} S(u,\, i) * \; Cr\big(p(u,\, i)\big)}{I(u)}$$

Those two models (RawMean & PeerTrust) were employed in our study for comparison purposes. Our experiments aim to illustrate the performance of the trust models compared to our new model (TrustMe model), and the preciseness of the produced trust scores. The comparison uses a straightforward metric, which is RawMean, and one more sophisticated model to conduct a fair comparison.

APPENDIX B. LIST OF PUBLICATIONS

Below is a set of four papers that I have participated in and been published during the course of this Ph.D. dissertation.

Paper 1:

Kendall E. Nygard, Md. Minhaz Chowdhury, Ahmed Bugalwi, Pratap Kotala, "People and Intelligent Machines in Decision Making", International Journal of Computers and Their Applications, 2017.

Paper 2:

Ahmed Bugalwi, Asaad Algarni and Kendall E. Nygard, A Trust Model for Bitcoin using Unsupervised Machine Learning, Proceedings of the 31st International Conference on Computer Applications in Industry and Engineering (CAINE 2018), New Orleans, 2018.

Paper 3:

Kendall E. Nygard, Ahmed Bugalwi, Maryam Alruwaythi, Aakanksha Rastogi, Krishna Kambhampaty, & Pratap Kotala. Elevating Beneficence in Cyberspace with Situational Trust. Proceedings of the 32nd International Conference On Computer Applications In Industry and Engineering (CAINE 2019), San Diego, 2019.

Paper 4:

Kendall E. Nygard, Ahmed Bugalwi, Maryam Alruwaythi, Aakanksha Rastogi, Krishna Kambhampaty, and Pratap Kotala, "Situational Trust and Reputation in Cyberspace", International Journal of Computers and Their Applications, 2019.