Applied Soft Computing 34 (2015) 693–704

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

GenTrust: A genetic trust management model for peer-to-peer systems☆

Ugur Eray Tahta[a,b], Sevil Sen[a,*], Ahmet Burak Can[a]

[a] Department of Computer Engineering, Hacettepe University, 06800 Ankara, Turkey
[b] ASELSAN, 06370 Ankara, Turkey

ARTICLE INFO

Article history:
Received 12 July 2014
Received in revised form 9 March 2015
Accepted 28 April 2015
Available online 4 June 2015

Keywords:
Evolutionary computation
Genetic programming
Trust models
Reputation
Peer-to-peer systems

ABSTRACT

In recent years, peer-to-peer systems have attracted significant interest by offering diverse and easily accessible sharing environments to users. However, this flexibility of P2P systems introduces security vulnerabilities. Peers often interact with unknown or unfamiliar peers and become vulnerable to a wide variety of attacks. Therefore, having a robust trust management model is critical for such open environments in order to exclude unreliable peers from the system. In this study, a new trust model for peer-to-peer networks called GenTrust is proposed. GenTrust has evolved by using genetic programming. In this model, a peer calculates the trustworthiness of another peer based on the features extracted from past interactions and the recommendations. Since the proposed model does not rely on any central authority or global trust values, it suits the decentralized nature of P2P networks. Moreover, the experimental results show that the model is very effective against various attackers, namely individual, collaborative, and pseudospoofing attackers. An analysis on features is also carried out in order to explore their effects on the results. This is the first study which investigates the use of genetic programming on trust management.

1. Introduction

Open nature of peer-to-peer (P2P) systems facilitates join or leave of users to the network without worrying about any obligations. However this freedom creates potential threats for good behaving peers. Since there is no central authority to manage inter-peer interactions, malicious peers can easily perform attacks or take advantage of system resources without contributing to the system. A way to mitigate such threats is to create artificial trust relationships among users based on peer interactions. Trust models can help in such open environments to quantify trustworthiness numerically and create trust relationships among peers. However, it is hard to measure and formulate trust with numeric values. Furthermore, measuring trust without a priori knowledge is a challenging problem in P2P systems since peers mostly interact with unknown peers. Therefore, trust management in P2P environments is a difficult research problem.

When there is a central authority, trust management is relatively easy problem. In some e-commerce applications, a central authority collects user inputs about completed interactions and this information is used to make trust decisions about future interactions. Although fake users and interactions can pollute the collected information, this model mostly works on e-commerce applications. However, P2P systems need more complex trust management models due to the lack of a central authority. Peers need to store and manage trust information about each other [1–3]. On the other hand, uncertain information collected from neighboring peers might be deceptive. Malicious peers might deliberately provide wrong information to the system and this might not be detected since there is no central authority. Therefore, trust models in P2P systems should be able to recognize various attacks and help benign peers to find trustworthy peers. While doing this task, ambiguous information collected from other peers should be processed carefully to make correct decisions.

The trust decision problem can be considered as a classification problem, and machine learning techniques could be employed to distinguish malicious peers from benign peers. This paper proposes a genetic programming (GP) based trust management model (GenTrust), extending our previous work [4] with greater experimental

☆ This paper is an extended, improved version of the paper "Evolving a Trust Model for Peer-To-Peer Networks Using Genetic Programming" presented at EvoComNet2014 and published in: Applications of Evolutionary Computing, Proceedings of 17th European Conference, EvoApplications 2014, Granada, Spain, April 23–25, 2014, LNCS 8602, Springer, 2014.

* Corresponding author. Tel.: +90 3122977500.
E-mail addresses: uetahta@aselsan.com.tr (U.E. Tahta), ssen@cs.hacettepe.edu.tr (S. Sen), abc@cs.hacettepe.edu.tr (A.B. Can).

http://dx.doi.org/10.1016/j.asoc.2015.04.053
1568-4946/

verification and analysis on features. The proposed model helps to identify malicious peers and find trustworthy peers using the features derived from peer interactions and recommendations. The model has evolved with these features by using genetic programming, which provides a mathematical function to measure trust values of peers. A peer ranks its neighbors according to trust values, and makes trusting decisions using these values. Each peer stores trust relationships for the peers they have interacted in the past. As peers gain more neighbors with time, malicious peers are excluded from the system using trust relationships. The evolved model is evaluated against various attackers, namely individual attackers, collaborators, and pseudospoofers. The results show that the model decreases the number of attacks considerably. Features of the model are also analyzed and their effects on the performance is assessed. Satisfaction related features are found more influential in trust decisions. Cross training and testing are performed among various attacker types to understand the model's adaptability on different attacker behaviors. These experiments show that the models trained on complex attack behaviors are also successful in simple attack behaviors.

Organization of the paper is as follows. Section 2 gives a summary of the state of the art research. Sections 3 and 4 explain the proposed trust model and the simulation environment respectively. Section 5 presents the experiments and discusses their results extensively. Section 6 outlines the conclusions of the study.

2. Related work

Trust is a social concept and hard to measure and formalize with theoretical foundations. Although some approaches have formulated trust as a result of direct experiences [5], most trust models use recommendations of others to build trust relationships [6–8]. However, it might be hard to correctly evaluate trustworthiness using recommendations, since recommendations may contain deceptive or subjective opinions [9]. To better address different aspects of the trust, some approaches use trust and distrust concepts [10,11], and some approaches formulate trust in different contexts [12,13].

Although it is hard to quantify trust numerically, reputation systems provide a means to address trust concept. As in some e-commerce applications, the users with higher reputation might be considered as more trustworthy. However, ensuring honesty of feedbacks in reputation systems is still a problem [14,15]. Dissemination of bogus feedbacks with multiple fake users (sybil attacks [16]) should also be prevented. Otherwise, malicious users can create many fake users and pollute reputation of others according to their intention [17,18]. Furthermore, users should also have long-lived identities to build higher reputations [14]. Otherwise, it is hard to maintain reputation when users join and leave the system frequently [19]. If reputation and trust concepts are modeled correctly, economic activity can be increased in e-commerce applications since some activities may not happen without trust [20].

Trust models have found many applications in P2P systems due to their open and malicious nature [21,22]. Since interactions mostly happen among unfamiliar peers, a trust model can improve the success rate of interactions and prevent some attacks [1–3]. Trust models on P2P systems are affected by the structure of network. In the unstructured overlay networks like Gnutella [23], trust information about other peers are obtained by flooding trust queries to the network [2,24,25]. Generally, each peer stores trust information about its neighbors. Trust queries enable to collect recommendations about unfamiliar peers and make decisions about them. Some trust models are designed based on structured P2P networks [1,3,26]. A distributed hash table (DHT), such as Chord [27], is used to manage trust information. The DHT algorithm determines which peer(s) will be in charge of storing the trust information about a peer. This provides efficient access to the global trust information without flooding queries to the whole network. However anonymity of the trust holders are revealed in this approach. Some models proposed cryptographic protocols to provide anonymity for trust holders [28].

Most trust models on P2P systems are generally based on probabilistic and statistical methods. Aberer and Despotovic [1] assume that number of complaints can be a measure of trustworthiness. Some approaches apply basic majority voting and averaging principles on the information collected from past experiences and recommendations, such as in XRep [29] or P2PRep [30]. EigenTrust model[3] uses transitivity of trust to calculate indirect trust relations. PeerTrust [26] uses transaction and community context parameters in trust calculation so application dependent factors and whole system related issues can be addressed better. Wang and Vassileva [31] use a Bayesian network model to evaluate different aspects of interactions on a P2P file sharing application. Selcuk et al. [24] use a vector-based trust metric and limited flooding approach to evaluate trustworthiness. PowerTrust [32] utilizes a random-walk strategy and power nodes in an overlay network to improve global reputation accuracy. GossipTrust [25] defines a randomized gossiping [33] protocol for efficient aggregation of trust values. Nguyen et al. [34] propose a Bayesian model of trust and use different context of trust but do not provide any experimental results. Conner et al. [35] proposes a method for customized trust evaluations by using different scoring functions over the same feedback data. Josang et al. [36] uses subjective logic to analyze trust networks. M-Trust [37] utilizes confidence in reputation for a better trust evaluation. Wu [38] proposes a trust model for predicting availability of wireless links on mobile P2P networks. A stable group model is proposed to address mobility issues and increase trust query success rate. Selvaraj and Anand [39] propose credential trees for evaluating trust among peers. In this model, peers use policies and credentials to make trust decisions. Anand and Bhaskar [40] integrates a trust model with a security model to solve some security issues of P2P systems. The model increases controlled scalability and availability of content in P2P systems. SORT [41] uses service and recommendation contexts of trust to measure trustworthiness better in providing services and recommendations.

Although most trust models use probabilistic and statistical methods, there are some approaches using machine learning techniques to classify good and malicious peers. Weihua Song et al. [42] use neural networks to derive trust values in multi agent systems. Neural network approach helps to classify recommendations as qualified or unqualified when choosing service providers. Beverly and Afergan [43] use a support vector machine based approach to select neighbors efficiently and then reduce the communication cost. Linear discriminant analysis and decision trees are used by Liu et al. [44] to help peers to build a knowledge base using past interaction history, which helps to identify successful transactions. Some approaches use Hidden Markov Models (HMM) in trust models [45–47] since HMM can be an effective method to model behaviors of entities in a system efficiently.

The evolutionary computation techniques are mainly applied to intrusion detection in the security domain [48,49]. Most of these studies focus on developing effective and efficient detection methods. Moreover they usually apply either genetic programming (GP) or genetic algorithms (GA). The first GP application for intrusion detection was from by Crosbie and Stafford [50]. Since then, there are many useful applications in the field. In [51], Abraham and Grosan compare the genetic programming technique with other machine learning methods for intrusion detection [51], and show that genetic programming techniques both outperform other techniques and are lightweight. Another advantage of evolutionary computation is to generate readable, easy-to-understand

outputs for security experts [52]. The grammatical evolution technique which produces readable grammars has been successfully employed for intrusion detection on wired networks [53] and on ad hoc networks [54]. Creating a set of solutions providing different trade-offs between conflict objectives is another characteristic of EC that attract researchers to investigate EC techniques on security. Sen and Clark [55] employed multi-objective evolutionary computation (MOEC) techniques in order to show how energy usage and detection ability can be traded off for resource-constrained networks. Moreover, there are recent studies which show the significant potential of evolutionary computation techniques to explore the suitable intrusion detection architecture, by taking into account the objectives of cooperative intrusion detection programs [55,56]. The MOEC techniques are also used to explore how intrusion detection system sensors could be best placed on a network in [57].

Although evolutionary computation techniques are increasingly used in the intrusion detection domain, as far as we know there is only one application of genetic algorithm in a trust model, which is proposed Selvaraj et al. [58]. This model uses genetic algorithm to detect attackers in a P2P system. Only features from local interaction data are used to make evaluations about trustworthiness. The trust model is trained by using the profiles of benign peers and, a profile base is created. If the behaviour of a peer deviates from this profile base, it is detected as malicious peer. This anomaly-based approach also uses a trusted central authority. However, GenTrust does not rely on a central authority. GenTrust uses features extracted both from interactions and recommendations. Thus both data from the peer's own experience and data collected from other peers are used. If a peer does not have enough information about another peer, it could use recommendations from its neighbor peers. The features and completely decentralized structure of GenTrust provide a more suitable model for P2P systems.

3. The model

Measuring trust without a priori knowledge is a challenging problem in P2P systems since peers mostly interact with unknown peers. In this research, GP is proposed to discover automatically complex properties of P2P networks. GP is a common evolutionary computation technique introduced by Koza [59]. Evolutionary computation (EC) is one of the most promising approaches in intrusion detection. A recent survey [49] showed that evolutionary computation techniques allow to obtain more readable outputs by security experts, lightweight solutions using fewer features, and a set of solutions providing different trade-offs between conflict objectives. Furthermore, EC provides ease of representation, and it does not require assumptions about the solution space [60]. Although EC techniques are increasingly applied to the intrusion detection problem, there is no application of GP in generating trust models. This is the first study that investigates the use of GP on trust management models on P2P systems without a central authority. While the problem in this study could also be represented by genetic algorithms (GA), we prefer to use GP over GA for its high usage and ease of representation on similar problems aiming to produce a metric.

In GP, an individual is represented with a GP tree, which is built of functions (operators, program statements etc.) and terminals (features, constants etc.). To find possible solutions to a problem, a group of individuals (populations), candidate solutions for the problem, are generated by GP in each generation. The first population is usually generated randomly. Subsequent generations are evolved by applying genetic operators such as crossover, mutation on individuals. Then, successful functions (individuals) are evolved in new generations to produce better possible solutions. The best solution to problem is sought by creating many generations. A fitness function is used to determine how well individuals create a solution to the problem. If the features and fitness functions are selected correctly, GP could produce more successful results than other machine learning techniques and programs written by people [61].

In this study, features extracted from past interactions and recommendations are used to build GP trees of individuals. Each tree provides a mathematical function to evaluate trust values. These functions are used when making trust decisions among peers. The peers with the highest trust values are selected for interactions in the network. The malicious peers which have low trust values are excluded intrinsically.

Table 1
Feature set of GenTrust.

Interaction Based Features	Recommendation Based Features
number of interactions *(int)*	number of recommendations *(rec)*
number of successful interactions *(sucInt)*	average of neighbors' number of successful interactions *(avgNeighbSucInt)*
average size of downloaded files *(avgFileSize)*	average of neighbors' average satisfaction values *(avgNeighbSat)*
average time difference between the last two interactions *(avgTime)*	average of neighbors' average weight values *(avgNeighbWeight)*
average weight *(avgWeight)*	average of trust values *(avgNeighbTrust)*
average satisfaction *(avgSat)*	

3.1. Feature sets and operators

Success of GP and other machine learning techniques is highly correlated to selecting the right feature set [62]. Selecting right features is a difficult problem since it is hard to determine at the beginning which features affect the result and help to distinguish various classes. As mentioned above, GenTrust uses features extracted from past interactions and recommendations of neighbors.

In GenTrust, an interaction is assumed to happen between two peers. The participants of an interaction keep some information about the results of the interaction. Any P2P application specific activity can be considered as an interaction, such as file sharing, CPU sharing, and storage sharing. In the simulations, a file sharing application is taken into account. Each peer extracts features using the information about past interactions. These features represent a peer's direct experience about its neighbors. Interaction based features are listed in Table 1.

Successful interactions are the interactions that the file download has finished successfully. Number of interactions and successful interactions provide an information about how much the other peer is known. For example, having only one interaction vs. having ten interactions do not represent equal level of experience. Average size of downloaded files represent how much a peer has contributed in each interaction. Average time difference between the last two interactions represent how frequently the other peer is interacted. Satisfaction and weight parameters are calculated as in [41]. Satisfaction parameter represents how good is the other peer in an interaction. In the simulations, it is assumed that peers make a bandwidth agreement before an interaction. Additionally, peers might change online and offline periods. Therefore, the satisfaction about an interaction is calculated based on average bandwidth,

Table 2
GP operators.

Operator	Symbol
summation	+
subtraction	–
division	/
multiplication	*
inverse	1/
log	rlog
square root	sqrt
square	square

agreed bandwidth before the interaction, online, and offline period values of the uploader:

$$Satisfaction = \begin{cases} (\frac{AveBw}{AgrBw} + \frac{OnP}{OnP + OffP})/2 & if \quad AveBw < AgrBw, \\ (1 + \frac{OnP}{OnP + OffP})/2 & otherwise \end{cases} \tag{1}$$

If an uploader provides the agreed bandwidth and does not go offline frequently, satisfaction value gets a value close to 1. Otherwise, it is assigned to a lower value between 1 and 0.

Some interactions might be more important than other interactions. For example, in a file download interaction, size of the file and popularity (number of uploaders) might affect the importance of a file. Weight parameter addresses the importance of interactions in GenTrust. Weight parameter is calculated based on the file size, the number of uploaders of the downloaded file, and the number of uploaders of the most popular file:

$$Weight = \begin{cases} (\frac{size}{100MB} + \frac{\#Uploaders}{Uploader_{\max}})/2 & if \quad size < 100\,MB, \\ (1 + \frac{\#Uploaders}{Uploader_{\max}})/2 & otherwise \end{cases} \tag{2}$$

If the size of a file is close to or larger than 100 MB and it is shared by many uploaders, the file is assigned to a larger weight value (close to 1). Thus downloading this file has more importance in trust calculations.

The second set of features are recommendation based features. When a peer wants to interact with another peer, it queries its own neighbors about their experiences, in other words recommendations. The neighbors who have information about the queried peer send their recommendations. A recommendation contains the following information: number of successful interactions, average satisfaction of interactions, average weight of interactions, and a calculated trust value of the queried peer. After a trust query is sent to neighbors, total number of collected recommendations and average of the values in collected recommendations are used as recommendation based features. These features are listed in Table 1.

The operators used in Gentrust are listed in Table 2. At the end of training step, a formula is generated for the trust calculation using these operators and the features listed in Table 1.

3.2. Fitness function

In evolutionary computation techniques, the fitness function is an important factor that affects classification performance. Success of a solution generated by an individual is determined by the fitness function. GenTrust uses the reduction in the number of attacks as the fitness function. To obtain a base case for the number of attacks, each experiment is run without having a trust model. This gives us an information about how many attacks occur in the simulated environment when a trust model is not present. Then this base is compared with the results obtained when a trust model exist. Let R_{trust} be the number of attacks with a trust model and $R_{noTrust}$ be the number of attacks without any trust model. Each interaction with a malicious peer (downloading a malicious file, or receiving unfair recommendations from the peer) represents an attack. In order to calculate the fitness function, the simulation module is executed for each individual. The fitness function is given below:

$$fitness = R_{trust}/R_{noTrust}. \tag{3}$$

Objective of the trust model is to decrease the value of fitness function. At the end of the evolution, the individual that minimizes the fitness function is selected as the solution. When an individual mitigates the number of attacks, the value of the fitness function decreases and then the success of the trust model increases. In other words, the individual resulting in the smallest number of attacks in the network is chosen. If the fitness function produces a value close to 1, this means that the model is not very effective and its use does not provide a benefit to the whole system. However, this does not mean that no peer is benefited from the model. Some peers might have benefited besides the whole system lacks.

4. Experimental settings

4.1. The attacks

In this section, the attack models considered in the study are outlined. Defining representative and realistic attack model is very important to evaluate a trust model successfully. The attacks covered here range from simple attacks to hard attacks in terms of detection. Attacks taking advantage of evasion strategies such as collaboration and pseudospoofing are also taken into account in the model.

In a P2P network, benign peers always behave as expected and properly carry out their tasks such as uploading authentic files, giving fair recommendations about the peers it has interacted with, and other similar means. On the other hand, a peer with a malicious intent could make severe damage to the network. A malicious peer could perform malicious activities such as uploading files with malicious content, giving biased recommendations about other malicious/benign peers. The aim of an effective model is to prevent malicious peers to participate in the network, hence to reduce the number of malicious files and unfair recommendations. In order to train an effective trust model and evaluate its performance, various attack scenarios are generated in this study. In the experiments, malicious peers are considered to behave in two different ways: naive and hypocritical.

- *Naive:* The attacker always uploads virus infected/inauthentic files and gives unfair recommendations to others [15].
- *Hypocritical:* The attacker performs attacks by uploading inauthentic files or giving unfair recommendations with x% probability. Otherwise, it acts like a good peer [3,24].

In addition to individual attackers, collaborators and pseudospoofers are also simulated in the experiments. Collaborators could behave naively or hypocritically as individual attackers. While individual attackers carry out their malicious activities on their own, collaborators perform attacks cooperatively. For example, they always uploads authentic files to each other and give unreasonably high recommendations about each other in order to protect his team friends from being identified. However collaborators may naively or hypocritically upload virus infected/inauthentic files and give unfair recommendations to other good peers.

The attackers discussed so far do not change their identities. In this study, another type of attack model, namely pseudospoofing, in which the attackers change their pseudonym in order to evade

being identified and remain active in the system for a longer period of time are analyzed. These attackers are shown to be the hardest to detect in the literature, due to periodical clearing of their negative reputations [19,24]. Selcuk et al. [24] state that the only way to exclude these attackers from the system is to build a sufficient level of trust among good peers. In this study, both individual pseudospoofers and collaborator pseudospoofers were analyzed. A pseudospoofer changes its identity every 1000 cycles in the experiments.

4.2. The simulation

The conceptual schema of the experiments is demonstrated in Fig. 1. First of all, the GP algorithm initializes a set of individuals randomly. Then each individual, the candidate trust formula, is simulated on a P2P network generated with a file sharing program in order to calculate the fitness function. The fitness value represents the number of malicious attacks reduced with the presence of the evolved trust formula. Based on fitness values, new individuals are created by applying genetic operators such as crossover and mutation on the existing population. This constitutes one generation of GP. The program is executed until the best individual is obtained or the defined number of generations are reached. Since the former one is difficult to obtain in a timely manner, the second termination criterion is mainly employed in GP applications. Since P2P simulation is executed for each individual evolved in the population, one run takes a very long time for the complex problem aiming to be solved in this research. Therefore the number of generations is fixed to be able to have many runs.

ECJ 21 toolkit [63] is employed for the GP implementation. In the experiments, the population and generation sizes are chosen empirically as 100 and 300 respectively. The other parameters employed are the default parameters of the ECJ toolkit. The GP algorithm is run ten times, and the best individual is selected among these runs. In the testing of the model, the best individual is run 30 times on a network. For the training and testing, each network setup (position of good and malicious peers, resources on a peer, peers requesting file downloads, etc.) is created randomly. The final results are obtained from these 30 runs. The general steps of the GP Module are listed in Algorithm 1.

Algorithm 1. The general steps of the GP algorithm.

```
initialize a random population
while current generation < = maximum generation do
  for all individuals in the current generation do
    execute the file sharing simulation
    evaluate the fitness function
  end for
  apply genetic operators (selection, crossover, mutation, etc.) to the individuals
  create new population
end while
```

In order to implement the attacker models described above, a file sharing simulation program is adapted from the file sharing simulator developed in [41]. To make observations realistic, simulation parameters are configured according to results of some empirical studies [64,65]. The simulation program is written in Java language. The GP module works in an integrated manner with the simulation module. The file sharing program is executed many times in order to simulate different types of attack models. Each run takes 50000 cycles and one cycle represents 10 minutes running of a P2P network. Each P2P network consists of 1000 peers. Based on the attack model, some of the peers are randomly assigned to be good or malicious peers. Peers simply implement the following activities: uploading files for sharing, interacting with other peers to download files, and giving recommendations about other peers when requested.

General flow of the file sharing simulation program is given in Algorithm 2. When the simulation is started, peers do not have any assumptions or information about others. All peers are strangers to each other. When a peer downloads a file from another peer, they become neighbors. Each interaction is saved for future interactions. The trust values are continuously built based on interactions they make with each other. When a file is requested, a list of file providers (*uploaders*) is returned by the P2P network. The trust values calculated by the GP model comes into the picture at this stage. The peer chooses the uploader with the highest trust value among neighbors or strangers offering the requested service. In the simulation model, a peer always choose neighbors over strangers. If none of the neighbors has the requested file, then a relationship is constructed with a stranger offering the file. Based on the interaction history with neighbors and recommendations collected from other peers, the peer chooses one of the providers to start downloading. When a download finishes, the downloader assigns a satisfaction value for the interaction and updates trust value of the uploader. If the downloaded file is malicious or inauthentic, the uploader is put into the blacklist and never be interacted again.

Algorithm 2. A high level algorithm on how simulation works.

```
1:  Initialize peers and resources
2:  while current cycle < maximum cycles do
3:    for each peer in the network do
4:      for each download of the peer do
5:        Update the status of the download
6:        if the download is completed then
7:          Update the trust value of the file provider
8:        end if
9:      end for
10:     if time to download a resource has come then
11:       Select a resource and send a query to find file providers
12:       Let U be a list of service providers
13:       if the peer has neighbors then
14:         if there are neighbors in U then
15:           Sort peers in U based on trust values
16:           Select the file provider with the highest trust value
17:         else
18:           for all peers in U do
19:             Send a reputation query to the neighbors
20:             Calculate a trust value
21:           end for
22:           Sort peers in U based on trust values
23:           Select the service provider with the highest trust value
24:         end if
25:       else
25:         Sort peers in U based on reported peer capabilities, e.g., bandwidth
27:         Select the service provider with the highest capabilities
28:       end if
29:     end if
30:   end for
31: end while
```

5. Experiments and analysis

5.1. Analysis on individual attackers

First of all, a trust model is generated for individual attackers. In the training, the generated individuals are simulated on a network in which 10% of the peers is malicious. The GP algorithm is run ten times for each attack type and the best results is evaluated in testing. Testing is done on networks with different amounts of malicious peers (10%, 30% and 50%). In the experiments, the attack probability of hypocritical attackers is set 20% for all interactions.

Table 3 shows the success ratio of the evolved trust model against individual attackers on different network setups. Here, malicious peers simply implement file-based attacks in which the attacker uploads a file with malicious content or an inauthentic file. Therefore, each network setup is executed both with the evolved trust model or without the trust model. The results represent the

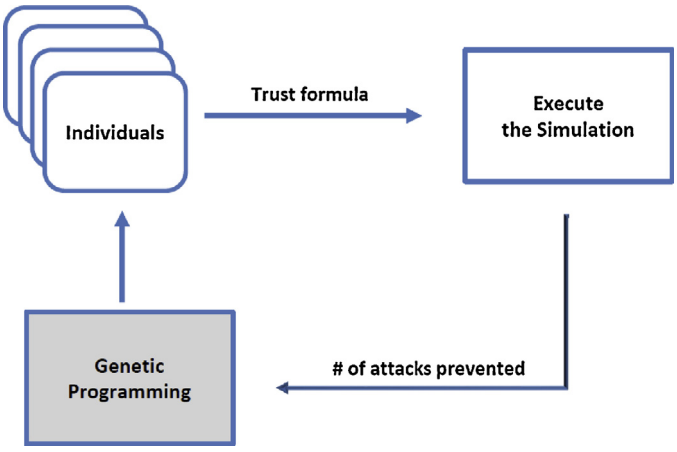

Fig. 1. Simplified schema of experiments.

Table 3
Success ratio of the trust model against individual attackers for the file-based attacks.

	10%	30%	50%
Naive	84.1 ± 0.6	77.3 ± 1.1	73.2 ± 0.8
Hypocritical	71.3 ± 1.2	58.3 ± 0.9	45.4 ± 1.3

amount of attacks reduced with the evolved GP model comparing to no trust model execution.

The evolved model identifies naive attackers successfully. As expected, the number of attacks is increased with the increase in the number of malicious peers in the network. However the model reduces the number of attack around 73% even in the case where half of the peers participated in the network is malicious. This good success ratio is a result of that individual naive attackers could easily be identified right after their first attack attempts. Hence their damage to the network is prevented enormously by excluding these peers from network activities immediately. The hypocritical individual attackers are also detected successfully. However they are more affected with the increase of malicious peers. The undetected or late detected attackers could affect the trust decisions in a negative way.

Identifying attacks in a reasonable time is related with convergence speed of the trust model. Fig. 2 shows the decrease in the number of attacks by naive and hypocritical individual attackers. The fast decrease in the attacks performed by naive individual attackers is also observed in this figure.

The evolved model is also evaluated against recommendation-based attacks in which malicious peers give unfair recommendations about good peers. The evolved trust model has also good performance on recommendation-based attacks. Fig. 3 shows the decrement in the recommendation-based attacks over time. In the model, if a peer intends to collect recommendations about another peer, it firstly requests recommendations from its trustworthy neighbors. Therefore, the unfair recommendation rate is mitigated over time as peers gain more neighbors. However, unfair recommendations do not drop as quickly as file-based attacks since determining an unfair recommendation is harder than determining an infected/inauthentic file.

5.2. Analysis on collaborative attackers

Secondly, the evolved model against collaborative attackers are assessed. These types of attackers are harder to detect, since they could protect each other from detection. In the experiments, the collaborators are worked in teams of 50 members. Again, the attack probability of hypocritical collaborators is set to 20% for all interactions in the experiments.

Table 4
Success ratio of the trust model against collaborators for the file-based attacks.

	10%	30%	50%
Naive	80.1 ± 1.2	73.9 ± 1.4	72.4 ± 0.8
Hypocritical	62.3 ± 0.7	45.6 ± 0.5	36.6 ± 1.0

Table 5
Success ratio of the trust model against individual pseudospoofers for the file-based attacks.

	10%	30%	50%
Naive	51.5 ± 1.2	46.8 ± 0.9	40.3 ± 1.1
Hypocritical	52.6 ± 1.4	45.2 ± 0.6	36.2 ± 0.8

The results are demonstrated in the Table 4. The performance of the evolved model against naive collaborators is as effective as the evolved model for naive individuals. Since naive collaborators are detected easily just after the first interactions with good peers, this attack model cannot take advantage of collaboration. They are detected long before other team members disseminate good recommendations about them. The performance of the model is quite effective even half of the network is constructed with malicious peers. The effect of collaboration is clearly seen on the results for hypocritical collaborators. In hypocritical behavior model, the collaboration helps malicious attackers to avoid being identified. Hence their identification becomes more difficult compared to hypocritical individuals. However the model still produces good results.

Fig. 4 shows the number of file-based attacks over time in a network consisting of 10% and 30% collaborators. The model dramatically decreases the number of effective attacks carried out by naive and hypocritical collaborators. The fast drop in the number of effective attacks by naive attackers is more clear as in the Fig. 2.

Since recommendation-attacks are only meaningful in the presence of a trust model, the performance of the evolved model cannot be compared with a network not building any recommendations. However the changes in the number of recommendation-based attacks in time are presented in Fig. 5. The number of attacks slightly increases when the collaboration takes place between peers compared to the Fig. 3. However, the trust model still mitigates the number of recommendation-based attacks. When sufficient information from interactions and recommendations are collected, the model starts to decrease the number of attacks. Slow, but a continuous decrease in the number of attacks is observed.

5.3. Analysis on pseudospoofers

In this section, the effectiveness of genetic programming on identifying individual pseudospoofers is evaluated. Pseudospoofers change their identities periodically, hence rebuild their neighbors and behave as a newly participated peer in the system. The GP model is trained with a P2P network consisting of 10% individual pseudospoofers. 10 simulations are run and the best individual among them is chosen. Three different networks settings with different levels of attackers (10%, 30%, and 50%) were used in the testing. In the experiments, the attack probability of hypocritical attackers was chosen as 20% for all interactions.

Table 5 shows the performance of the model against naive and hypocritical pseudospoofers. While the detection ratio of naive pseudospoofers was 51.5%, the model presents slightly better performance on preventing from hypocritical pseudospoofers (52.6%) in the network. The model also shows reasonable results, even in P2P networks in which half of the peers behave maliciously. A big

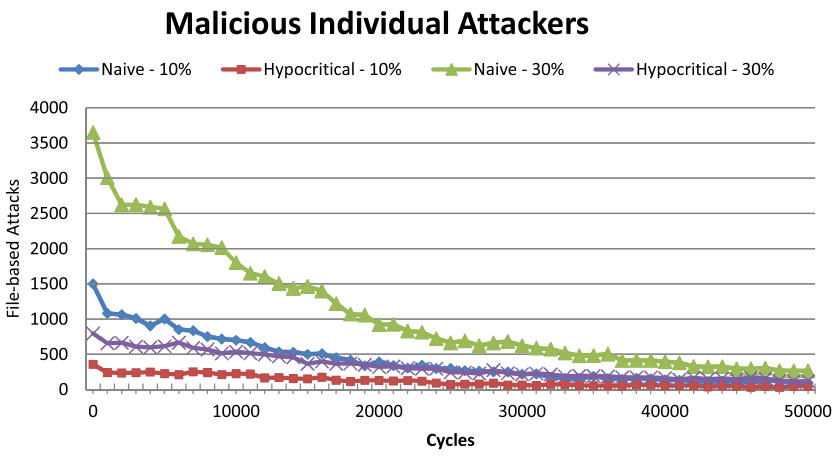

Fig. 2. File-based attacks over time in a network consisting of individual attackers.

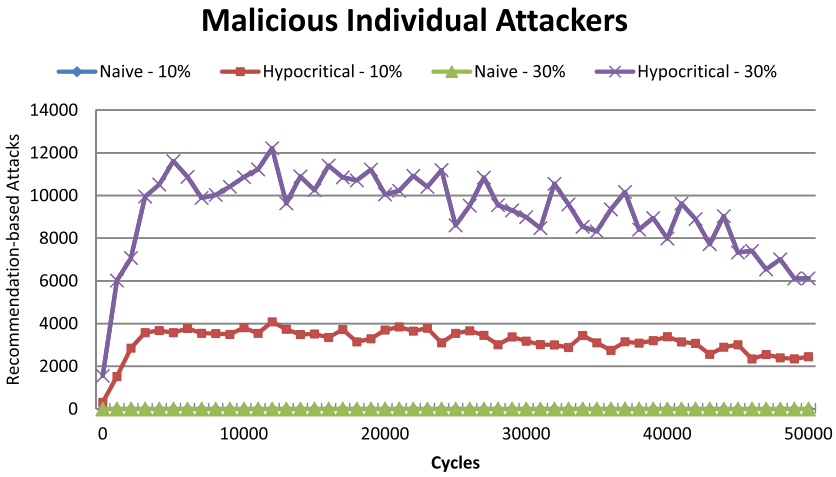

Fig. 3. Recommendation-based attacks over time in a network consisting of individual attackers.

decrease in the identification of naive attackers was observed when they changed their identities. Since naive attackers send infected files continuously, they are the easiest to detect in normal circumstances. However in this situation, they could evade recognition by changing their identities and successfully removing their bad history. On the other hand, hypocritical pseudospoofers remove not only their bad history, but also good reputations as a result of successful interactions with good peers. Therefore, their identification ratio is slightly higher than for naive pseudospoofers, which is quite different from the results in the previous sections. While there is a dramatic difference in the success ratios of naive and hypocritical attackers in the previous sections so far, it is not the case in the pseudospoofing attack model due to its very nature.

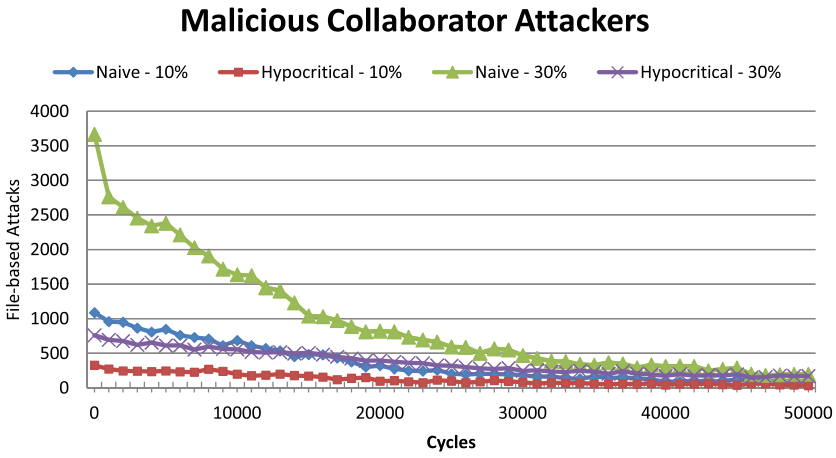

Fig. 4. File-based attacks over time in a network consisting of collaborators.

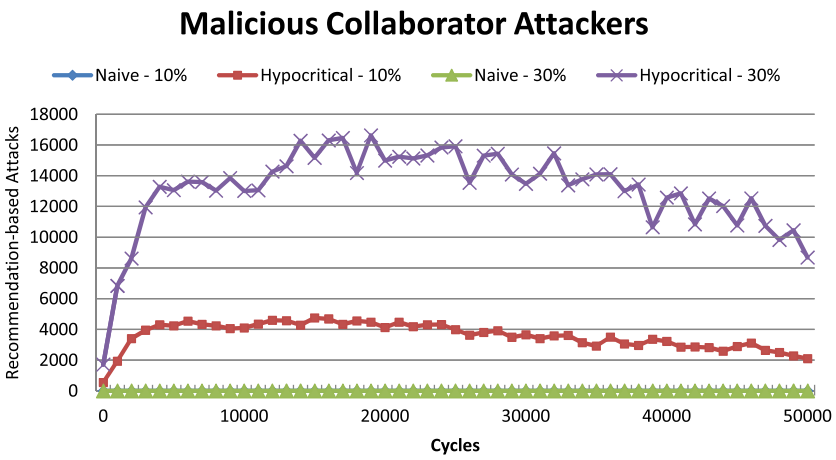

Fig. 5. Recommendation-based attacks over time in a network consisting of collaborators.

Table 6
Success ratio of the trust model against collaborator pseudospoofers for the file-based attacks.

	10%	30%	50%
Naive	51.1 ± 0.8	46.1 ± 0.7	36.4 ± 1.1
Hypocritical	50.4 ± 1.0	43.6 ± 1.2	30.7 ± 1.4

Table 7
Success ratio of the trust model in mixed environment for the file-based attacks.

	30%	50%
Mixed Attackers	68.3 ± 0.5	52.1 ± 1.1

5.4. Analysis on collaborative pseudospoofers

In this section, the pseudospoofing model was evaluated with collaborative attackers. The team size of collaborators was set to a maximum of 50 peers. The GP model was again trained with a network in which 10% of the peers were malicious. The performance of the model is demonstrated in Table 6. As it is shown, attacking collaboratively does not make a big difference for both naive pseudospoofers and hypocritical pseudospoofers. It could be concluded that changing identities of attackers is more effective than attacking collaboratively in order to evade detection.

5.5. Analysis on mixed malicious environments

After performing tests on each attacker type, the success of the model was tested on mixed attacker environments. Models are trained and tested on environments containing different types of attackers in different concentrations. Training is performed on environments containing 30% attackers from all types mentioned in the previous sections. Testing is done on 30% and 50% malicious environments with different types of attackers. During the tests, a collaborative team consisted of 50 members, attack probability of a hypocritical attacker is 20% and the identity changing period is set to 1000 periods for pseudospoofers.

The success ratios of the model of mixed malicious environments are given in Table 7. As shown in the table, the model was very successful with 68.3% and 52.1% success ratios in 30% and 50% malicious environments respectively. Peer-to-peer systems in the real world may contain different types of malicious users. The results in this experiment show that the model produces realistic and acceptable results in the environments close to real world conditions. The model is able to learn different behaviors of different attackers and mitigate attacks in such extremely malicious environments. Fig. 6 shows the number of file-based attacks for 1000 periods in a mixed malicious environment. At first, good peers have difficulty to identify malicious peers in such a mixed environment. However, they are able to identify and isolate malicious peers over time, as they start to interact with more peers.

To understand the model's improvement with respect to number of generations, the model was tested on a 30% mixed malicious network with a different number of generations. As shown in Fig. 7, the model's prevention ratio for file-based attacks increase as more generations are used. Genetic operations on larger populations aid to train better trust models to mitigate file-based attacks.

Table 8
Success ratio of cross training and tests between naive and hypocritical individuals.

		TEST	
		Naive	Hypocritical
TRAINING	Naive	82.4 ± 0.4	53.6 ± 0.7
	Hypocritical	81.6 ± 1.1	70.9 ± 1.3

5.6. Cross training and tests among various attacker types

In the previous tests, training and tests are done on the same type of attackers. In this section, training and testing was conducted between different types of attackers. This helps to understand adaptability of a trained model against an unknown attacker type. Models trained on naive attackers were tested on hypocritical attackers and vice versa. Training and testing in these experiments were done using 10% malicious network setups. Attack probability of hypocritical attackers was set to 20%.

In the first experiment, the models trained on naive and hypocritical individuals tested among each other. As shown in Table 8, the model trained on naive attackers mitigated 53.6% of the attacks when tested on a network with hypocritical attackers, which is approximately 17% lower than the model trained with hypocritical attackers. The model trained with simple attack behavior (naive) was not very successful on complex hypocritical attack behavior. Because the model trained for simple attack behavior produces a simple solution which may not be sufficient to mitigate attacks in complex environments. However, the model trained on hypocritical behavior produced 81.6% success ratio on the naive attacker environment, which is only 1% less than the model trained with naive attackers. Hence the model trained on more stealthy, complex attack behavior can be successful on simpler attack behaviors.

In the second step of experiments, more difficult collaborative attack behavior was compared to individual attack behavior. In

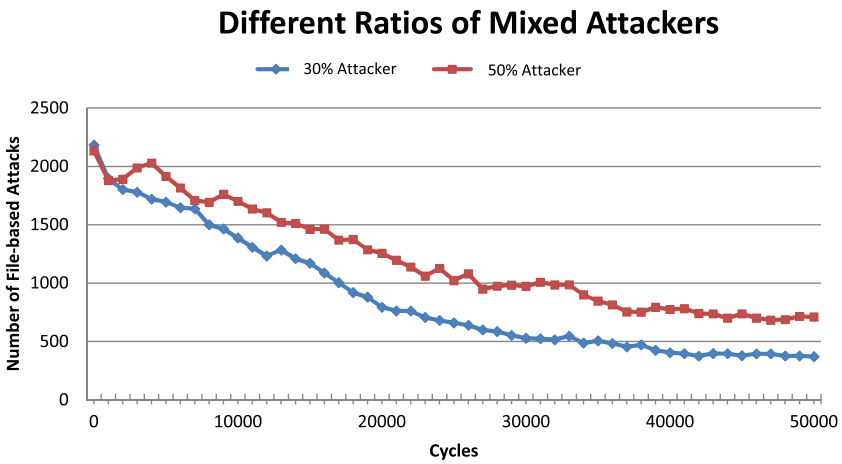

Fig. 6. File-based attacks over time in an environment with different types of attackers.

Table 9
Success ratio of cross training and tests between hypocritical individuals and hypocritical collaborators.

		TEST	
		Individual	Collaborator
TRAINING	Individual	70.6 ± 1.0	49.3 ± 0.9
	Collaborator	67.3 ± 1.2	60.2 ± 0.8

these experiments, only hypocritical attack behavior was considered since it is more difficult than naive behavior. Table 9 shows the results of these experiments. As observed in the previous experiment, the model trained on more complex behavior (hypocritical collaborator) produces good results on simple behavior (hypocritical individuals). However, the model trained on hypocritical individuals produced a 49.3% success ratio on hypocritical collaborators, which is nearly 11% less than the model trained on hypocritical collaborators. These results verify that the attack resistance of a model can increase in the training phase according to the difficulty and complexity of the attacks.

5.7. Feature analysis

The choice of features used for machine learning is very important. These features must contain sufficient information to allow the fundamentals to be developed. However too many or irrelevant features could degrade the performance of GP. In this section, several experiments are run in order to investigate the effects of satisfaction, weight, and recommendation features on the performance of the model. These features were used in both the interaction-based and recommendation-based feature sets. The attack models employed in the experiments are given as follows: naive attackers, hypocritical attackers, naive collaborators, hypocritical collaborators. In the first experiment, the model is trained using all features given in Table 1, except the weight related features. Secondly, only satisfaction related features are excluded from the feature set. Lastly, recommendation features in Table 1 are not included in the training. 10% malicious network setups are employed in both training and testing. The team size of collaborators is set to a maximum of 50 peers, and the attack probability of hypocritical attackers set to 20%, as in previous experiments. The results of these experiments are shown in Table 10.

The results show that all features are critical in the identification of malicious peers. Satisfaction related features seem more influential than both weight and recommendation related features on the results. Since the results of interactions, successful or not, directly affects the satisfaction related features, these features plays an important role in the recognition of attacks.

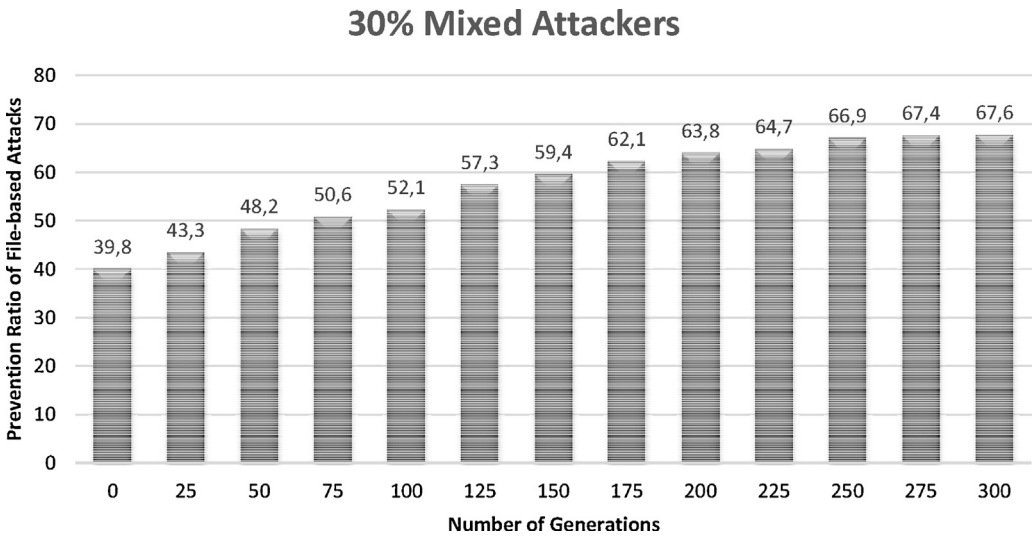

Fig. 7. Prevention ratio of file-based attacks in a 30% malicious environment with respect to various generation numbers.

Table 10
Effects of features on GenTrust performance for file-based attacks.

	Satisfaction Feat. excluded	Weight Feat. excluded	Recommendation Feat. excluded	All Feat.
Naive Attackers	74.7 ± 0.6	77.1 ± 0.7	82.6 ± 1.2	84.1 ± 0.6
Hypocritical Attackers	54.6 ± 1.0	57.3 ± 0.8	59.2 ± 0.5	71.3 ± 1.2
Naive Collaborators	71.2 ± 0.9	71.8 ± 0.7	78.9 ± 1.1	80.1 ± 1.2
Hypocritical Collaborators	48.8 ± 1.1	51.6 ± 1.0	51.7 ± 1.3	62.3 ± 0.7

Table 11
Evolved formulas for individual naive and collaborative hypocritical attackers.

Attack Type	Evolved Formula
Individual Naive	(rLog(avgSat + (avgNeighbWeight − (avgNeighbSucInt % avgNeighbSat))) * (sqrt(sucInt / avgFileSize) * (avgSat + (avgNeighbWeight − rLog(avgNeighbSat * avgNeighbSucInt)))) * sqrt(avgNeighbTrust / sucInt)) / sqrt((avgSat +avgNeighbWeight) * avgNeighbSucInt)
Collaborative Hypocritical	((((avgNeighbSucInt − avgNeighbWeight) − avgNeighbWeight) + avgFileSize / (avgNeighbSucInt / (sucInt * sqrt(avgNeighbSat))) / ((sqrt(avgSat) + ((avgNeighbWeight + (avgNeighbSucInt − sqrt(avgSat))) / sqrt(avgNeighbTrust)))))) / sqrt(avgSat))) * (avgNeighbTrust % sqrt(sucInt * (avgNeighbWeight − rLog(avgNeighbSat)))) * rlog(sqrt(avgSat − (avgNeighbSat / sqrt(avgNeighbWeight))))

Excluding some features especially decreases the performance of the model against hypocritical attacks. Nonetheless, the evolved model against naive attackers still produces reasonable results even when lacking of weight related or satisfaction related features during training. The absence of recommendation related features shows a negligible decline in naive attackers. Naive attackers/collaborators are identified in their first attempts before building relationships with their neighbors. Thus the model evolved for this type of attackers was not affected from absence of some features as much as the model for hypocritical attackers/collaborators. Since hypocritical attackers has more sophisticated attack behavior, absence of recommendation related features could negatively affect the performance of GenTrust. If a peer interacts with a hypocritical attacker while he is behaving benignly, it may not identify the attacker. In such cases, the only way to identify such attackers is to get recommendations about the malicious peer from other peers.

The examples of evolved formulas for individual naive attacks and collaborative hypocritical attacks are given in Table 11. These formulas are used to calculate trust values of each candidate uploaders, then the uploader with the highest trust value is selected for an interaction. As it can be seen from these formulas, GenTrust uses most of the features in the evolved individuals. Another observation is that the more complex attack behavior has produced a more complex formula.

5.8. Comparison and discussion

Since trust models in the literature have different assumptions and simulation environments, it is difficult to compare our results with other works. Peer properties, attacker models, and network assumptions (DHT, unstructured, or hybrid) are not same among the papers in the literature. For example, Eigentrust [3] uses a DHT based storage model and assumes the existence of pretrusted peers who help in establishment of trust relations. Similarly, Peertrust [26] uses a DHT based storage. Powertrust [32], uses power nodes and a DHT structure to leverage trust establishments. Another work proposing a genetic algorithm based trust model [58], uses a centralized peer profile base, which is completely different than our distributed architecture. In addition to such assumptions, attacker and network models in these works are not same as GenTrust. Since such assumptions have great impact on trust evaluation, it is hard to compare the results of these works with GenTrust. Even assumptions are similar, there is no standard simulation environment in the literature. The studies in the literature generally use their own simulation code, which is generally not accessible for comparisons.

Table 12
Success ratio of SORT and GenTrust against individuals for the file-based attacks.

		10%	30%	50%
Naive	SORT	83.5 ± 0.8	79.1 ± 1.2	75.9 ± 0.9
	GenTrust	84.1 ± 0.6	77.3 ± 1.1	73.2 ± 0.8
Hypocritical	SORT	71.4 ± 0.7	59.5 ± 1.1	42.4 ± 1.3
	GenTrust	71.3 ± 1.2	58.3 ± 0.9	45.4 ± 1.3

Table 13
Success ratio of SORT and GenTrust against collaborators for the file-based attacks.

		10%	30%	50%
Naive	SORT	81.1 ± 1.0	78.7 ± 0.9	74.2 ± 1.2
	GenTrust	80.1 ± 1.2	73.9 ± 1.4	72.4 ± 0.8
Hypocritical	SORT	62.2 ± 0.6	38.4 ± 1.4	22.7 ± 0.8
	GenTrust	62.3 ± 0.7	45.6 ± 0.5	36.6 ± 1.0

For more consistent comparisons, the models should be tested on the same simulation environment with the same attacker and network models and similar peer population assumptions.

In this paper, GenTrust is compared with SORT trust model [41]. Since SORT has similar simulation environment and assumptions, we were able to simulate it for the same experiments. Table 12 gives the results of SORT and GenTrust on individual attacker experiments. GenTrust produced comparable results with SORT. GenTrust has performed slightly better than SORT in the worst case scenario of individual attacker experiments (in 50% malicious network with hypocritical attackers). This shows that GenTrust has stable performance in the extremely malicious environments.

Another experiment with SORT is performed on collaborative attackers. Table 13 gives the results of SORT and GenTrust on collaborative attackers. As it is expected, success of SORT decreases in the collaborative attacks especially for hypocritical ones. GenTrust performed better than SORT for the hypocritical attacker model. Especially in the hard scenarios of the hypocritical collaborators (30% and 50% malicious networks), GenTrust provided a more robust performance than SORT. This shows that the proposed model for hypocritical collaborators is successful on handling extremely malicious environments.

6. Conclusion

This paper proposes GenTrust, a trust model evolved using genetic programming. The generated model allows each peer to calculate trust values of other peers based on interaction and recommendation based features. Naive and hypocritical attacker models are studied with individual, collaborative, and pseudospoofing

behaviors. The model is trained against these attack types and evaluated on various network setups containing different ratio of malicious peers. The experimental results show that the model could successfully decrease different types of attacks. Naive attackers are identified easily in both individual and collaborator scenarios. Hypocritical attackers are more difficult to deal with and more successful when they collaborate. Pseudospoofing complicates the identification of malicious peers as expected. The evolved trust model has decreased the number of file-based attacks with promising success ratios in most scenarios. Recommendation-based attacks are mitigated but do not decrease as much as file-based attacks, due to the difficulty of recognizing misleading recommendations. In cross training and tests, the models trained with simple attacker behaviors were not very successful on complex behaviors. However the model trained on complex attacker behavior showed good performance on simple attackers, which were not included in the training. This shows that the model has a durable performance on different and new attack behaviors. The importance of features on the performance are also explored. Satisfaction related features are more influential than both weight related and recommendation related features, since they directly represent successful/unsuccessful interactions. Especially, identification of hypocritical attackers/collaborators are badly affected in the absence of these features.

Overall, the evolved model showed that genetic programming could successfully be employed to build a trust model in peer-to-peer networks. Genetic programming has a considerable potential for exploring the complex design spaces associated with trust management on P2P networks. The proposed model suits to the decentralized nature of P2P networks, since it does not rely on any central authority or global trust values. In the future works, this model can be extended further with online learning.

References

[1] K. Aberer, Z. Despotovic, Managing trust in a peer-2-peer information system, in: Proceedings of the 10th International Conference on Information and Knowledge Management (CIKM), 2001.
[2] F. Cornelli, E. Damiani, S.D.C. di Vimercati, S. Paraboschi, P. Samarati, Choosing reputable servents in a p2p network, in: Proceedings of the 11th International World Wide Web Conference, 2002.
[3] S. Kamvar, M. Schlosser, H. Garcia-Molina, The (eigentrust) algorithm for reputation management in P2P networks, in: Proceedings of the 12th World Wide Web Conference (WWW), 2003.
[4] U. Tahta, A. Can, S. Sen, Evolving a trust model for peer-to-peer networks using genetic programming, in: Proceedings of EvoCOMNET, Vol. 8602 of LNCS, 2014, pp. 3–14.
[5] S. Marsh, Formalising trust as a computational concept, Ph.D. thesis, Department of Mathematics and Computer Science, University of Stirling, 1994.
[6] A. Abdul-Rahman, S. Hailes, Supporting trust in virtual communities, in: Proceedings of the 33rd Hawaii International Conference On System Sciences (HICSS), 2000.
[7] B. Yu, M. Singh, A social mechanism of reputation management in electronic communities, in: Proceedings of the Cooperative Information Agents (CIA), 2000.
[8] L. Mui, M. Mohtashemi, A. Halberstadt, A computational model of trust and reputation for e-businesses, in: Proceedings of the 35th Hawaii International Conference On System Sciences (HICSS), 2002.
[9] A. Jøsang, E. Gray, M. Kinateder, Analysing topologies of transitive trust, in: Proceedings of the First International Workshop on Formal Aspects in Security and Trust (FAST), 2003.
[10] R. Guha, R. Kumar, P. Raghavan, A. Tomkins, Propagation of trust and distrust, in: Proceedings of the 13th International Conference on World Wide Web (WWW), 2004.
[11] P. Victor, C. Cornelis, M. De Cock, P. Pinheiro da Silva, Gradual trust and distrust in recommender systems, Fuzzy Sets Syst. 160 (10) (2009) 1367–1382.
[12] D. Gefen, Reflections on the dimensions of trust and trustworthiness among online consumers, SIGMIS Database 33 (3) (2002) 38–53.
[13] G. Swamynathan, B.Y. Zhao, K.C. Almeroth, Decoupling service and feedback trust in a peer-to-peer reputation system., in: Proceedings of the 2005 International Symposium on Parallel and Distributed Processing and Applications (ISPA), Vol. 3759 of LNCS, 2005.
[14] P. Resnick, K. Kuwabara, R. Zeckhauser, E. Friedman, Reputation systems, Commun. ACM 43 (12) (2000) 45–48.
[15] C. Dellarocas, Immunizing online reputation reporting systems against unfair ratings and discriminatory behavior, in: Proceedings of the 2nd ACM conference on Electronic commerce, EC '00, 2000.
[16] J. Douceur, The sybil attack, in: Proceedings of the First International Workshop on Peer-to-Peer Systems (IPTPS), 2002.
[17] H. Yu, M. Kaminsky, P.B. Gibbons, A. Flaxman, Sybilguard: defending against sybil attacks via social networks, SIGCOMM Comput. Commun. Rev. 36 (4) (2006) 267–278.
[18] N. Tran, B. Min, J. Li, L. Subramanian, Sybil-resilient online content voting, in: Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation (NSDI), 2009.
[19] E.J. Friedman, P. Resnick, The social cost of cheap pseudonyms, J. Econ. Manag. Strateg. 10 (2) (2001) 173–199.
[20] Z. Despotovic, K. Aberer, Trust-aware delivery of composite goods, in: Proceedings of the 1st International Conference on Agents and Peer-to-peer Computing, 2002.
[21] T. Bearly, V. Kumar, Building trust and security in peer-to-peer systems, in: Secure Data Management in Decentralized Systems, Springer, 2007, pp. 259–287.
[22] L. Mekouar, Y. Iraqi, R. Boutaba, Reputation-based trust management in peer-to-peer systems: taxonomy and anatomy, in: Handbook of Peer-to-Peer Networking, Springer, 2010, pp. 689–732.
[23] N. Stakhanova, S. Ferrero, J.S. Wong, Y. Cai, A reputation-based trust management in peer-to-peer network systems, in: Proceedings of 16th International Conference on Parallel and Distributed Computing Systems (ISCA), 2004, pp. 510–515.
[24] A.A. Selcuk, E. Uzun, M.R. Pariente, A reputation-based trust management system for p2p networks, in: Proceedings of the 2004 IEEE International Symposium on Cluster Computing and the Grid, CCGRID '04, 2004.
[25] R. Zhou, K. Hwang, M. Cai, Gossiptrust for fast reputation aggregation in peer-to-peer networks, IEEE Trans. Knowl. Data Eng. 20 (9) (2008) 1282–1295.
[26] L. Xiong, L. Liu, Peertrust Supporting reputation-based trust for peer-to-peer ecommerce communities, IEEE Trans. Knowl. Data Eng. 16 (7) (2004) 843–857.
[27] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, H. Balakrishnan, Chord: A scalable peer-to-peer lookup service for internet applications, ACM SIGCOMM Comput. Commun. Rev. 31 (4) (2001) 149–160.
[28] A. Visan, F. Pop, V. Cristea, Decentralized trust management in peer-to-peer systems, in: IEEE 2011 10th International Symposium on Parallel and Distributed Computing (ISPDC), 2011, pp. 232–239.
[29] E. Damiani, D.C.D. Vimercati, S. Paraboschi, P. Samarati, F. Violante, A reputation-based approach for choosing reliable resources in peer-to-peer networks, in: Proceedings of the 9th ACM Conference on Computer and Communications Security, 2002.
[30] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, P. Samarati, Managing and sharing servents' reputations in p2p systems, IEEE Trans. Knowl. Data Eng. 15 (4) (2003) 840–854.
[31] Y. Wang, J. Vassileva, Bayesian network trust model in peer-to-peer networks, in: Proceedings of 2nd Workshop on Agents and Peer-to-Peer Computing at the Autonomous Agents and Multi Agent Systems Conference (AAMAS), 2003.
[32] R. Zhou, K. Hwang, Powertrust: A robust and scalable reputation system for trusted peer-to-peer computing, IEEE Trans. Parallel Distrib. Syst. 18 (4) (2007) 460–473.
[33] S. Boyd, A. Ghosh, B. Prabhakar, D. Shah, Randomized gossip algorithms, IEEE/ACM Trans. Netw. 14 (SI) (2006) 2508–2530.
[34] G.H. Nguyen, P. Chatalic, M.-C. Rousset, A probabilistic trust model for semantic peer to peer systems, in: Proceedings of the 2008 International Workshop on Data Management in Peer-to-Peer Systems, ACM, 2008, pp. 59–65.
[35] W. Conner, A. Iyengar, T.A. Mikalsen, I. Rouvellou, K. Nahrstedt, A trust management framework for service-oriented environments., in: Proceedings of World Wide Web Conference, 2009.
[36] A. Jøsang, R. Hayward, S. Pope, Trust network analysis with subjective logic, in: Proceedings of the 29th Australasian Computer Science Conference, 2006.
[37] B. Qureshi, G. Min, D. Kouvatsos, A distributed reputation and trust management scheme for mobile peer-to-peer networks, Comput. Commun. 35 (5) (2012) 608–618.
[38] X. Wu, A distributed trust management model for mobile p2p networks, Peer-to-Peer Networ. Appl. 5 (2) (2012) 193–204.
[39] C. Selvaraj, S. Anand, A role based trust model for peer to peer systems using credential trees, Int. J. Comput. Theory Eng. 3 (2) (2011) 234–239.
[40] P. Rubesh Anand, V. Bhaskar, A unified trust management strategy for content sharing in peer-to-peer networks, Appl. Math. Modell. 37 (4) (2013) 1992–2007.
[41] A.B. Can, B. Bhargava, Sort: A self-organizing trust model for peer-to-peer systems, IEEE Trans. Dependable Sec. Comput. 10 (1) (2013) 14–27.
[42] W. Song, V.V. Phoha, X. Xu, An adaptive recommendation trust model in multiagent system., in: Proceedings of IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2004), 2004.
[43] R. Beverly, M. Afergan, Machine learning for efficient neighbor selection in unstructured p2p networks, in: Proceedings of the 2nd USENIX workshop on Tackling Computer Systems Problems with Machine Learning Techniques, SYSML'07, 2007.
[44] X. Liu, G. Tredan, A. Datta, A generic trust framework for large-scale open systems using machine learning, Comput. Intell. 30 (4) (2014) 700–721.
[45] M.E.G. Moe, M. Tavakolifard, S.J. Knapskog, Learning trust in dynamic multiagent environments using HMMS, in: Proceedings of the 13th Nordic Workshop on Secure IT Systems (NordSec 2008), 2008.

[46] E. ElSalamouny, V. Sassone, M. Nielsen, HMM-based trust model, in: Proceedings of the Workshop on Formal Aspects in Security and Trust, Vol. 5983 of LNCS, 2010, pp. 21–35.
[47] X. Liu, A. Datta, Modeling context aware dynamic trust using hidden Markov model, in: Proceedings of the Conference on Artificial Intelligence (AAAI), 2012.
[48] S. Wu, W. Banzhaf, The use of computational intelligence in intrusion detection systems: A review, Appl. Soft Comput. 10 (2010) 1–35.
[49] S. Sen, A survey of intrusion detection systems using evolutionary computation, in: X.-S. Yang, S.F. Chien, T.O. Ting (Eds.), Bio-inspired Computation in Telecommunications, Elseiver, 2015, pp. 73–92, Chapter 4.
[50] M. Crosbie, G. Stafford, Applying genetic programming to intrusion detection, in: Proceedings of Symposium on Genetic Programming in Conference on Artificial Intelligence (AAAI), 1995.
[51] A. Abraham, C. Grosan, Evolving intrusion detection systems, in: Proceedings of Genetic Systems Programming: Theory and Experiences, Vol. 13, 2006, pp. 57–79.
[52] A. Orfila, J. Tapiador, A. Ribagorda, Evolving high-speed, easy-to-understand network intrusion detection rules with genetic programming, in: Proceedings of EvoWorkshops on Applications of Evolutionary Computations, Vol. 5484 of LNCS, 2009, pp. 93–98.
[53] D. Wilson, D. Kaur, Knowledge extraction from kdd'99 intrusion data using grammatical evolution, WSEAS Trans. Inf. Sci. Appl. 4 (2007) 237–244.
[54] S. Sen, J.A. Clark, A grammatical evolution approach to intrusion detection on mobile ad hoc networks, in: Proceedings of the Second ACM Conference on Wireless Network Security (WiSec), 2009.
[55] S. Sen, J. Clark, Evolutionary computation techniques for intrusion detection in mobile ad hoc networks, Comput. Netw. 55 (15) (2011) 3441–3457.
[56] A. Hassanzadeh, R. Stoleru, On the optimality of cooperative intrusion detection for resource constrained wireless networks, Comput. Secur. 34 (2013) 16–35.
[57] H. Chen, J.A. Clark, J.E. Tapiador, S.A. Shaikh, H. Chivers, P. Nobles, A multi-objective optimisation approach to ids sensor placement, in: Proceedings of the Conference on Computational Intelligence in Security for Information Systems, 2009.
[58] C. Selvaraj, S. Anand, Peer profile based trust model for p2p systems using genetic algorithm., Peer-to-Peer Netw. Appl. 5 (1) (2012) 92–103.
[59] J.R. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press, Cambridge, MA, USA, 1992.
[60] D. Fogel, What is evolutionary computation? IEEE Spectr. 37 (2000) 28–32.
[61] W. Banzhaf, F.D. Francone, R.E. Keller, P. Nordin, Genetic programming: an introduction: on the automatic evolution of computer programs and its applications, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1998.
[62] M.A. Hall, Correlation-based feature selection for machine learning, Department of Computer Science, University of Waikato, 1999, Ph.D. thesis.
[63] Ecj 21, A java-based evolutionary computation and genetic programming research system, 2013 http://www.cs.umd.edu/projects/plus/ec/ecj/
[64] S. Saroiu, P. Gummadi, S. Gribble, A measurement study of peer-to-peer file sharing systems, in: Proceedings of the Multimedia Computing and Networking, 2002.
[65] S. Saroiu, K. Gummadi, R. Dunn, S.D. Gribble, H.M. Levy, An analysis of internet content delivery systems, in: Proceedings of the 5th USENIX Symposium on Operating Systems Design & Implementation (OSDI), 2002.