Computer Standards & Interfaces 44 (2016) 94–101

Contents lists available at ScienceDirect

Computer Standards & Interfaces

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

Patching by automatically tending to hub nodes based on social trust

Xin Liu, Yao Wang, Dehai Zhao, Weishan Zhang, Leyi Shi

College of Computer and Communication Engineering, China University of Petroleum (East China), Qingdao, China

ARTICLE INFO

Article history:
Received 31 January 2015
Received in revised form 29 June 2015
Accepted 3 August 2015
Available online 8 August 2015

Keywords:
Social trust
Social computing
Automatic patching
Hub nodes
Vulnerability

ABSTRACT

Malicious code can propagate rapidly via software vulnerabilities. In order to prevent the explosion of malicious codes on the Internet, a distributed patching mechanism is proposed in which the patch can tend to hub nodes automatically based on social computing in social networks. A server in social network generates automatic patches and then selects those nodes with maximum degree to push automatic patches to. Those hub nodes then send the patch to their buddies according to their degree in social network. Automatic patches propagate rapidly through hub nodes and patch nodes in social network, which will improve the security of the whole social network. Those receivers accept the patch according to trust value to the sender, which can avoid some malicious codes exploit our scheme to propagate themselves. Experiments show this mechanism is more efficient than other patching mechanisms.

1. Introduction

According to the "2013 National Information Security, Computer and Mobile Terminal Virus Epidemic Survey Analysis Report," published by National Computer Virus Emergency Response Center, 76.4% network security incidents were caused primarily by malicious codes, and vulnerabilities in system and network without patching are still serious [1].

Malicious codes generally propagate by three kinds of methods: software vulnerability, users, or a combination of the two methods. Some malicious codes can start automatically without user involvement, such as worm, embedded script, etc. In order to propagate more effectively, most worms do not destroy their host during the propagation. The host may not realize it has been infected in general. For those host systems and applications with vulnerabilities, patching the system or the application is very effective. That is to say, those hosts which are vulnerable or have already been infected can download and install corresponding patches against the worms.

Here is an example of vulnerability.

Vulnerability MS06-014 is a logical vulnerability [2]. There is a vulnerability in RDS.Dataspace ActiveX, which binds with Microsoft Data Access Components (MDAC), leading to remote code execution vulnerability because it may not guarantee security interaction in certain conditions. An attacker who successfully exploits this vulnerability can take complete control of the host system.

Exploiting vulnerability MS06-014 will not occupy a lot of system memory and will not lead to browser crash and implement anti-antivirus easily, which makes it is one of the most influential vulnerability. There are lots of web trojan generators exploiting this vulnerability with a high success rate.

Exploit.MS06014.c is a script virus, which propagate itself by exploiting the vulnerability MS06-014. It generally propagates by web malicious codes. If the patch corresponding to MS06-014 has not been installed in a host, the host will be infected by this code when the user browses the webpage containing this malicious codes. Then the user host is under remote control by an attacker in all probably.

According to statistics from China National Vulnerability Database of Information Security (CNNVD), the number of vulnerabilities is always on the uptrend. Installing the security patches in OS and application program is an effective implement to patch network hosts. However, many users are unwilling install patches, or they forget or ignore to install the patches. Thus, the automatic patching mechanism is very necessary to protect user hosts. In this paper, we proposed a patching scheme in which automatic patch propagates in social network and patch the vulnerable hosts on the Internet to improve the security of hosts and the Internet.

The remainder of the paper is organized as follows. Section 2 presents related work. In Section 3, we explicate the motivation for automatic patching. The patching mechanism automatically tending to hub nodes based on social computing is explicated in Section 4. We presents security analyses and experimental evaluation in Sections 5 and 6. We conclude and present some future work in Section 7.

2. Related work

2.1. Automatic patching mechanism

Vojnovic and Ganesh researched the validity of automatic patching mechanism [3], taking the dissemination speed of the patches utilized in worm containment into quantization. An automatic patching system should be able to detect worms, generate patches, disseminate patches, verify and install patches. The author verified that using filter together with patch can improve effectiveness of worm containment radically.

http://dx.doi.org/10.1016/j.csi.2015.08.001
0920-5489/

Xie and Zhu utilized existing P2P overlay network structure to disseminate security patches to susceptible hosts automatically [4]. It took two measures. One was based on segmentation, utilizing immune hosts to prevent the propagation of worms in overlay network in advance. The other one was based on connected dominating set (CDS), utilizing a group of dominating nodes in the overlay network to improve the dissemination speed of patches.

Shakkottai and Srikant researched two different dissemination strategy for defense against worms propagation [5]. First, they defended against worms incomparably with a fixed number of patching server. However, it generally took no time for worms to infect a large number of hosts, so that patching server with a fixed number could not deal with worm propagation. Second, they utilized PULL mechanism to disseminate patches to P2P nodes. Each P2P node randomly connected to another node and then inquired whether there is the patch or not. If there was, then the node downloaded, verify, and installed it. It could restrain the worm propagation effectively by using the exponential patching dissemination speed in P2P network.

Friedman proposed a method in which users scan friends' machines to make them defend users' local host [6]. Although the third-party scanning was a kind of intrusion to the local host, the scanning results were valuable and trustworthy. If there was a copy of the patch in a security host, then this host could scan its neighbors so that the susceptible neighbors made fences to propagate the patch to all hosts in which vulnerabilities exist. That is, a node which had installed patches could patch its neighbors. The system had two modes such as critical mode and casual mode. The critical mode was based on the new patches, embedding release date in records of the patches. The casual mode was to patch the minority hosts which had not been patched.

Zhu proposed a strategy that can contain phone worms in early stage of propagation [7]. It could establish a social relationship graph, a presentation of the most probable propagation path of phone worms, by the analysis of network flow. It used two algorithms to segment the social relationship graph and then chose the phone user who can infect the most phone users as the best phone set that should be patched first so that it could slow down the phone worm propagation speed and narrow the propagation range.

Stelios Sidiroglou and Angelos D. Keromytis proposed an architecture for automatically repairing software flaws that are exploited by zero-day worms [8]. This approach relies on source code transformations to quickly apply automatically created patches to vulnerable segments of the targeted application.

2.2. Social trust

Lazer et al. described that a computational social science is emerging that leverages the capacity to collect and analyze data with an unprecedented breadth and depth and scale [9]. The use of social trust relationships is both practical and necessary as the Web evolves [10].

Social computing is used in multiple fields, such as recommender systems [11][12], spam filtering [13][14], limiting free-riding in P2P networks [15], P2P routing [16], defending against Sybil attacks [17], defending against malicious Web pages [18].

When users share and exchange information with other people in the OSN, they have to manage the risk involved with the transactions without previous experience and knowledge about other users reputation. One way to solve this problem is to establish the system that can help users assign trust values to unknown users. Trust is the foundation of all interactions among society members [20].

Golbeck verified that the most correct information is from the most credible node in her doctoral dissertation by experiments [19]. She proposed TidalTrust, which is an algorithm for inferring trust, considering the trust values to be numbers in a continuous range [0,1]. The author showed that trust values inferred through shortest path may be more accurate. This is a classical algorithm with high citation rate, which is compared with by many algorithms later to prove efficiency.

We proposed a method of forming secure P2P network using benign worms against malicious worms [21]. The benign worm with a hitlist is generated to patch and clean the corresponding malicious worm for peers in the P2P network, which is based on the largest distance list. The benign worm propagates along the hitlist. The spread of the benign worm is also a distributed patching process. The patch can be disseminated more quickly, and the network congestion is much less than centralized patching scheme in P2P networks.

SocialTrust [22] provided community users with dynamic trust values by distinguishing relationship quality from trust, incorporating a personalized feedback mechanism for adapting as the community evolves and tracking user behavior.

Xin et al.[23] proposed a dynamic trust conference algorithm for social network. When there is only one shortest path from the source node to the destination node, the trust value calculated by the algorithm of Golbeck [19] will be the same as the trust value of the node, which is the last but one in the chain to the destination node. The result is not rational because a user usually has less trust in a stranger than a friend. In such case, the trust value to the destination node is calculated as the product of the trust values of the nodes adjacent to each other in the trust train. Then the authors give the mechanism for calculating dynamic trust value according to users' behavior in the network.

Zhuo et al.[24] introduced time-based dynamic trust model (TDTM), a model for time-based dynamic trust. Every node in the distributed environment is endowed with a trust vector, which figures the trust value between this node to the others. The trust value is dynamic due to the time and the inter-operation between two nodes. This change is quantified based on the mind of pheromone in the ant colony algorithm.

Yan et al. [25] proposed an algorithm for trust cluster head election based on ant colony systems. The cluster head plays an important role in clustering wireless sensor networks (WSNs). In [26], the authors presented a trust model for WSNs, called BTRM-WSN, based on ant colony systems. This model uses the mind of the ant colony system to find the most trustworthy route to the node that provides the requested service. The node in the route is selected according to the pheromone on the edge. The pheromone of a trace is regarded as the amount of trust on the edge and is used for the process of cluster head election. The goal of ACO algorithm in the model is to find the most trustworthy path to a request node. ACO is not used for trust calculation, which is not mentioned in the paper.

Bedi et al. [27] proposed a trust-based recommender system in which using ant colony for trust computation. The process of the trust computation is not the same as that in this paper. The thinking of the ant colony is used to select best neighborhood for the active user in the recommender system. The best neighborhood of the active user is a specific number of most trustworthy nodes that is linked to it. Along with those neighbors' ratings about some items, the active user can make better decisions.

3. Motivation

User nodes with various operation systems and applications in social network scatter around the Internet. Each software has its own particular vulnerability, which results in the special security problems of nodes in social networks. In order to improve the defending ability against malicious codes, nodes with vulnerabilities need to be patched.

Nodes on the Internet usually belong to some overlay networks such as P2P network, Instant Messaging network (IM network), and social network. Some special vulnerability exists in particular client software. For P2P network, many P2P worms propagate automatically exploiting vulnerabilities that exist in P2P software itself. It results in most peers in the P2P network are vulnerable, which makes some methods cannot carry out. For example, it is difficult to utilize the P2P network topology to contain the worms because most peers are vulnerable [28]. How to make nodes on the Internet be healthy?

Social computing is useful to resolve this problem because social network is so prevalent that most nodes on the Internet are in certain social networks. Messages diffuse so fast in social network that the Internet become a small world. Some nodes in the scale-free network have a large number of connections, called Hub nodes. The degree of nodes in social network follows power-law distribution. Hub nodes in social network have more contacts than normal nodes, which can make they have some advantages over other nodes in information propagation. If we can patch the Hub nodes first, they can push automatic patches to much more users than normal nodes. In order to install security patches in vulnerable hosts with users' authorization, we can make the patch propagate by human computing in social network through hub nodes, which can improve the defending ability of overlay networks and the Internet against malicious worms.

4. Patching mechanism automatically tending to hub nodes based on social trust

A user generally uses more than one Instant Messaging tools and has multiple IM accounts. Users need more convenient and more secure communication with their friends by various IM tools. To take advantage of users' friendship and resources fully in protection system, it is better to integrate various IM tools together so that users' buddy list in various IM tools can be managed as an unit. That is to say, we can integrate QQ, MSN, Skype, Alitalk, and Fetion together as a synthetic IM tool.

Each instant messaging network is an online social network. Various instant messaging networks can be integrated to a larger complex social network, called synthetic instant messaging network, which can cover most of the Internet users. In a user host, various buddy lists of multiple IM are integrated as a unified buddy list which make the number of users' trustworthy friends increase, so that users can take advantage of more security experience and information from more friends to resist malicious codes.

The process of patching vulnerable nodes in social network is described as follows. The topology of the social network and the social relationships among users are stored in a database in the server of the social network. A node with its direct and indirect friends in synthetic IM network form the egocentric network of this node. The egocentric network of node S in synthetic IM network is shown in Fig. 1.

The server in social network can calculate the number of links for each node and select hub nodes to push a patch to. Then automatic patches can launch the file sending function of the various IM tools to send patches to the friend node who has more connections than other friend nodes in the user's buddy list, so that the automatic patches in local host can be pushed to next nodes and spread around the social network along the egocentric network of each node and patch vulnerable nodes in IM network.

Whether the receiver accepts the patch sent to it according to his trust to the sender, we set a reasonable trust threshold to restrict the acceptance of a patch on receivers. If the trust value to the sender reach the trust threshold, the receiver can accept the patch. Otherwise, the receiver should refuse to accept it. The trust threshold can avoid some malicious codes and disguised patch exploit our scheme to propagate on the Internet.

There are two kinds of patches, automatic installed patch and automatic propagation patch, which are generated by the security server.

4.1. Composition of automatic patches

Automatic installing patch can be installed only in local host when it is pushed to the node and cannot propagate itself in the network. The server downloads the corresponding patches to the vulnerabilities from official patch server, according to the information of nodes in the server, and then embeds the autorun function into the downloaded patches to generate the normal automatic installing patches, as shown in Fig. 2.

The automatic propagating patch can implement functions such as auto-download, auto-install, and auto-propagate. We can embed auto-send function in the automatic installing patch to generate automatic propagating patch, as shown in Fig. 2.

We can implement the auto-download function by the Push technology that the sender automatically sends patches to the client. The server pushes automatic patches to the vulnerable nodes by advanced strategy. After the user hosts in social network received an automatic patch, the automatic patches will be installed automatically in local host. Then the patch launches the file sending function of IM to push itself to the user's friends whose trust value is over the threshold in the unified buddy list. By this way, the patch can propagate automatically. The server and hosts send patches by push technology during the propagation, so that patches can spread around the social network automatically and instantly.

According to the IM clients' software and the system platform on the host, the automatic patch adds different codes to implement automation function. It needs different codes to automatically launch the program in different platform. For example, it needs different codes to implement the auto-installing function for Windows and Linux. It also needs the ability that can launch file sending function in different kinds of IM tools to implement the automatic propagation function.

4.2. ACO-based trust inference

Malicious codes may disguise an automatic patch to exploit our scheme to propagate in the social network. We utilize the trust between

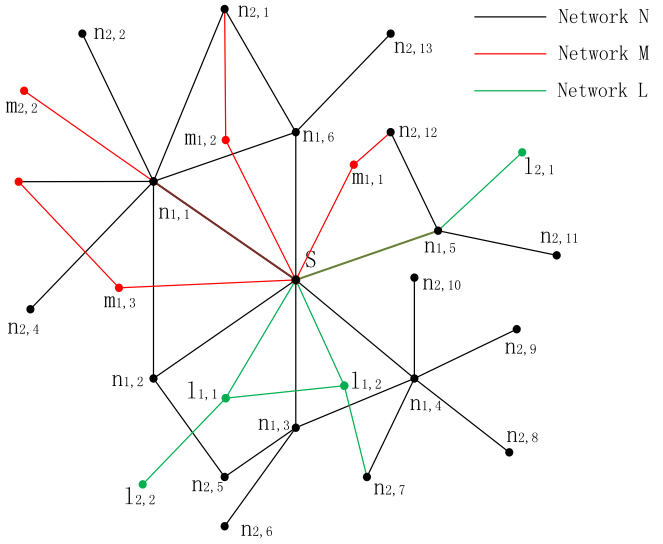

Fig. 1. The egocentric network of S in synthetic IM network.

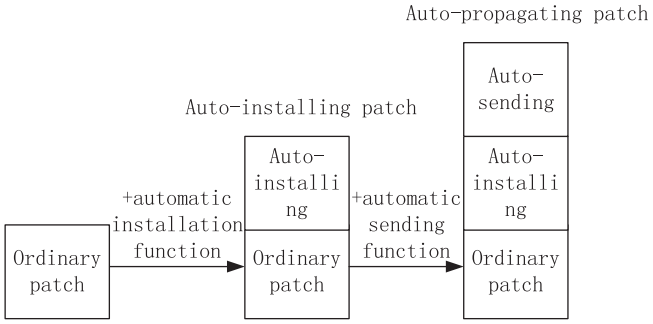

Fig. 2. The composition of the automatic patch.

two users to avoid a user receiving malicious codes from his contacts. That is, a user only accepts automatic patches from his trusty contacts. A user assigns the trust value of all his friends in his unified buddy list at the client of IM tools, and the trust value of the indirect friends is calculated on the host. If a user sends a malicious code to his friends, and those friend nodes find the malicious code, they will set the trust value to the sender to 0. The trust value will be changed according to interactions between user nodes. The dynamic trust value of each friend is calculated according to the friend's contribution to the security of user host, which can avoid the damage wherein a friend, which disguises as a benign node for a long time, unexpectedly sends malicious codes or malicious messages to the user.

A user trusts his friends, and the friends trust their friends too. Then the user can trust his indirect friends by his direct friends' recommendation, that is to say, trust can be transmitted, as shown in Fig. 3. For example, suppose user i trusts user j highly and user j trusts user k highly, then user i would trust user k almost as highly. Moreover, if user j does not trust user k, user i would does not trust user k as user j does. That is to say, the transmittal of trust value T_{jk} from user j to user i relies on trust value to user j, where T_{jk} is the trust value for user j to user k. By contrast, if user i does not trust user j, then whether user j trusts user k or not, user i would be uncertain about user k.

In a user's egocentric network, a trust chain can be formed by the center node as the initial node and several friend nodes of the user. The degree of trust is fewer with the trust chain growth [29]. There are several chains among trusted users, showed in Fig. 1. The number of nodes in a trust chains is not exceeding 3 in general. If the trust value is high enough, there will be long trust chains, such as a chain from $n_{2,1}$ to $n_{2,7}$: $n_{2,1} \rightarrow n_{1,1} \rightarrow S \rightarrow n_{1,4} \rightarrow n_{2,7}$, which does not belong to egocentric network of node S. The longer the chain , the more the available resources.

According to trust chains from the center node to each node, the indirect trust value to each node can be calculated.

If a contact is a direct friend of a user, the direct trust value is adopted with neglecting other trust chains between the user and his friend.

If the destination node is not the direct friend of the initial node, a user can obtain the trust value to his indirect friend by the transmission and calculating of trust value. The user can accept files and messages from indirect friends with higher trust value, so that the experience of indirect friends can be utilized by the user.

By data analysis, Golbeck represented that the most accurate information would come from the most trusted neighbors. She set a threshold for trust value and proposed a model for referring trust value, TidalTrust [19], which was applicable to distributed systems. If the trust value of direct contact attains the threshold, the corresponding path can be adopted to calculate recommendation trust value. The formula for calculating the recommendation trust value of indirect friend is showed in (1).

$$T_{is} = \frac{\sum_{j \in N(i),\ T_{ij} \geq T_{\min}} (T_{ij} \times T_{js})}{\sum_{j \in N(i),\ T_{ij} \geq T_{\min}} T_{ij}} \tag{1}$$

where T_{ij} represents the trust value for user i to user j, and $N(i)$ represents the set of all friends of user i, which includes direct and indirect friends.

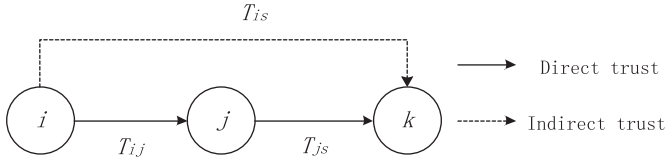

Fig. 3. The transmission of the trust.

If there is only one recommendation node in the path, the result $T_{ik} = T_{jk}$ can be obtained using Eq. (1), which indicates the indirect trust for user i to user k is equal to the trust value for user j to user k. The result is not rational because the stranger has less trust in a user than the user's friend. We use ant colony optimization algorithms to select two paths with different length to avoid the serious deviation. After all, there are generally not only one path between two nodes in social network.

The ant colony optimization (ACO) algorithm was initially proposed by Marco Drongo in his PhD thesis [12]. The ACO algorithm is a probabilistic technique for solving computational problems, which can be reduced to finding good paths through graphs. It imitates the behavior which ants seeking a path between their colony and a source of food [3]. In natural world, ants initially wander randomly and upon finding food return to their colony while laying down pheromone trails. If other ants find such a path, they are likely not to keep travelling at random but to instead follow the trail, returning and reinforcing it if they eventually find food. When searching for an optimal path in a graph, we can imitate the behavior of the ants, leaving pheromone after one optimal path is found. The pheromone is important for later searching process. Let $T_{ij}(t)$ be the intensity of trail on edge (i, j) at time t. After each ant finds one optimal path at time $t + n$, the trail intensity can be updated according to the following formula:

$$T_{ij}(t+n) = \rho \cdot T_{ij}(t) + \Delta T_{ij} \tag{2}$$

where ρ is a coefficient ($0 \leq \rho \leq 1$) that represents the evaporation of some pheromone in the trail between time t and $t + n$, imitating the feature of human memory that new information stored while the old faded away.

$$\Delta T_{ij} = \sum_{k=1}^{m} \Delta T_{ij}^{k} \tag{3}$$

where ΔT_{ij}^{k} is the quantity of trail substance laid on edge (i, j) by the kth ant between time t and $t + n$. It is given according to specific problem.

The definition of transition probability is as follows:

$$p_{ij}^{k} = \begin{cases} \dfrac{[T_{ij}(t)]^{\alpha} \cdot [\eta_{ij}(t)]\beta}{\sum_{S \subset \text{allowed}_k} [T_{is}(t)]^{\alpha} \cdot [\eta_{is}(t)]\beta} & j \in \text{allowed}_k \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where p_{ij}^{k} represents the probability that ant k in node i choose node j as the next node in the path. T_{ij} is the element of the trail intensity matrix T, while $T_{ij}(t)$ represents the intensity of trail on edge $(i; j)$ at time t, and η_{ij} is heuristic function for the edge $(i; j)$, which indicates the ant in the system is not completely blind. α and β are parameters that control the relative importance of trail versus heuristic function.

For calculating the trust value to an indirect node, we need to find the trust chains and decide which chain to choose by the ant colony optimization algorithm which is an optimization method fitted to find the minimum cost in graph optimization problems. Thus, our goal is to find the path by which we can get high trust value to destination node. For an ant k, to find an appropriate path can be divided into iterated single steps as which node can be chosen as the next one. The probability that ant k in node i chooses node j is given by Eq. (4).

The elements of matrix T are set to the same constant during the initial process of the algorithm and are updated every time an ant stops its searching process according to the trust value calculated through the trust chain. Here we set parameter η as the direct trust value of node i to node j, which helps the ant to choose the node with high trust value. The set allowed$k$ is the taboo list, and it acts as the ant's memory, keeping the list of nodes that the ants have already visited and that cannot be chosen again.

According to the six degrees of separation theory [13], we assume that the trust chain consists of no more than six nodes to increase the algorithm efficiency. When the trust chain that found by an ant consists of more than six nodes, we can let this ant stop. Besides, users would not exchange information with the contacts with lower trust. Thus, a threshold $T_{\min}$ is needed to avoid a node accepting a disguised patch. The ant would stop when the current trust value calculated through this chain is lower than $T_{\min}$ during the process of finding the trust train.

After we get two trust chains to the destination node, the trust value can be calculated by Eq. (1).

According to the trust value we get, we can update the trail intensity of the edges in the trust trains we select using Eq. (2).

ΔT_{ij}^{k} in Eq. (3) can be defined as

$$\Delta T_{ij}^{k} = Q \cdot T_{ij} \tag{5}$$

where Q is a constant, representing the intensity of the trail. It can influence the speed of the convergence of the algorithm. In such case, edges through which we can get high trust value will gain more pheromone.

4.3. Patch Pushing Strategy

The server or a node in social network pushes a patch to its neighbors. However, the order of nodes pushed to is according to the neighbor node's degree. The larger the node's degree, the earlier the node is chosen. The node with higher degree can push the patch to more neighbor nodes than the node with lesser links, which makes the patch propagate faster.

4.3.1. Theorem

The propagation time of an automatic patch pushed earlier by a node with higher degree is shorter than pushed earlier by a node with lower degree.

A patch propagates in a social network with N nodes. According to the node degree from high to low, nodes are marked by 1, 2, 3, … , N. Let n_1, n_2, …, n_N represent the degree of corresponding nodes, then $n_1 \geq n_2 \geq \ldots \geq n_N$. Set the time from a node starting to push a patch to another node to the patch received by the receiver fully is t. The time when node i receive the patch is t_i. Then $t_i = t + n \times t$, where n is the number of nodes the patch has gone across before it arrives at the node.

Nodes in social network can form multi-communities, which include multiple hub nodes. The sever pushes a patch to hub nodes according to the degree, which makes all communities are patched. The highest degree nodes first receives the patch.

Let T_i be the time that all direct neighbor nodes of node n_i received the patch, which can be calculated by Eq. (3):

$$T_i = (i + \ldots + n_i + n_i + 1 + \ldots + n_i + i{-}1)t \tag{3}$$

T_i^k is the time that all direct neighbor nodes of node n_i received the patch later for kt.

$$T_i^k = (k + i + \ldots + n_i + (n_i + 1) + \ldots + (n_i + k + i{-}1))t \tag{4}$$

The node i does not receive the patch at $n_i t$ but at $(n_i + k)$t, where k is an integer. If $k \geq 0$, the node receives the patch later for kt. If $k < 0$, the node receives the patch earlier for kt. Thus,

$$\begin{aligned}\Delta_i^k &= T_i^k - T_i \\ &= (k + i + \ldots + n_i + (n_i + 1) + \ldots + (n_i + k + i{-}1))t \\ &\quad -(i + \ldots + n_i + (n_i + 1) + \ldots + (n_i + i{-}1))t \\ &= kn_i t\end{aligned}$$

Then

$$\begin{aligned}&\left(T_i^k + T_j^{-k}\right) - (T_i + T_j) \\ &= \left(T_i^k - T_i\right) + \left(T_j^{-k} - T_j\right) \\ &= kn_i t + \left(-kn_j t\right) = k(n_i - n_j)t\end{aligned}$$

Therefore, $(T_i^k + T_j{}^{-k}) - (T_i + T_j) \geq 0$ □

This indicates that node 1 is the 2nd node the server pushed the patch to while node 2 is the 1st node the server pushed the patch to. The time spent by two nodes pushing the patch to all their neighbors is longer than the case in which the patch is pushed to nodes according to the node degree.

The time elapsed through the patch propagation can be calculated by Eq. (5)

$$T = \sum_{i=1}^{N} t_i \tag{5}$$

where T is the time elapsed through the patch propagation in the whole social network. The sequence of nodes patched is different from the mark sequence. The server pushes the patch to nodes according to the order number of nodes, which makes the nodes with more neighbors receive patch earlier than nodes with less neighbors.

The server pushes patches to the node with the maximum degree, and then automatic patch continues to push itself into the node's friends in the unified hubby list in different IM networks. According to the IM network where this user's friends are, the patch propagates itself along multiple paths in multiple social networks which consist of multiple kinds of IM tools.

4.4. Process of automatic patching propagation

The server downloads a patch and forms an automatic propagating patch. Automatic patches propagate itself along the egocentric network of each node in the synthetic IM network, from one node of the trust chain to another node which is in another trust chain.

For example, the server pushes automatic patches to node S, and then automatic patches propagate itself along the egocentric network of S. S_1 and S_2 are the direct friends of S, and both of them have their own egocentric networks. After S pushes automatic patches to S_1 and S_2, automatic patches will continue to propagate itself along multiple trust chains of the egocentric network of S_1 and S_2, as shown in Fig. 4.

Automatic patches for vulnerable peers in overlay network are pushed to nodes of the IM network to continue to propagate. These nodes of IM network are also nodes of the overlay network. The IM tool notices user to receive the file after a friend node in the social network has sent an automatic patch to it. If the user determines the trust value to the sender reach the trust threshold, it will accept the patch. Each node that has accepted the automatic patch in the social network continues to send the patch to all its friends. The patch is automatically installed in local host. If the user node has already been patched or is not vulnerable, the propagation will be halted. If the local host detect the malicious code, it will set the trust value to the sender to 0 and send the trust value to the sender to all its friends. Then its friends will calculate the indirect trust value to the sender. Those friend nodes will not accept the patch because the lower trust value to the indirect sender.

The server maintains social data of nodes and relationships in social network and updates the data periodically. The server recalculates the degree for each node after each updating and chooses vulnerable nodes to push the corresponding automatic patch according to the propagation strategy.

The automatic patching needs users' involvement during the propagation of automatic patches, which increases the security of user hosts; however, it also slows down the speed of the patch propagation. We

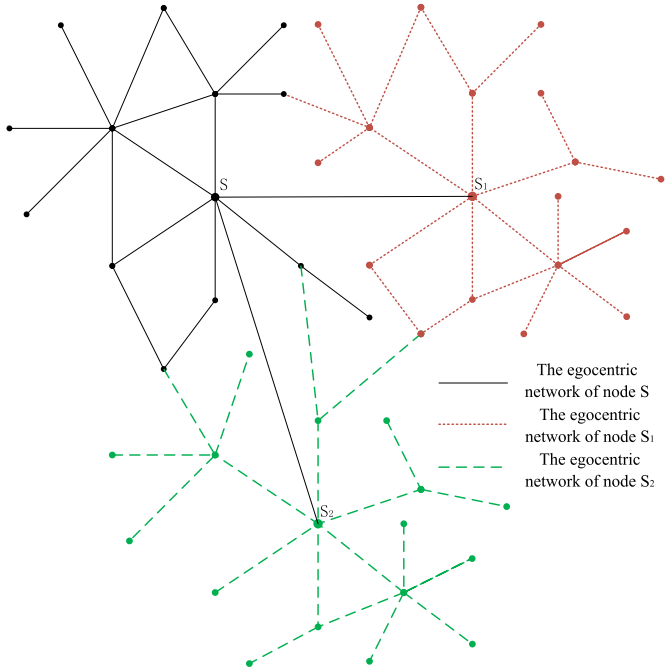

Fig. 4. The propagation of the patch in the social network.

utilize IM tools as social network platform, which is beneficial from the instantly interaction between users and make patches propagate speedily.

5. Security analyses

In this mechanism, a user is directly involved in the receiving of an automatic patch. The user subjectively decides whether to accept this patch according to the trust value to the sender, when the user receives the request of sending automatic patch from the friend. If a malicious user exploits this mechanism to send malicious codes to others, the anti-virus software of the receiver checks the patch. If the anti-virus software detects that the patch is malicious, the trust value to the sender will be set to 0. The indirect contacts of the sender will also be distrust the sender because the trust value calculated according to the recommendation from the direct receiver. There are various anti-virus software products on the Internet. That is to say, the probability a disguised patch can be detected by various anti-virus software products will be improved, which can propagate in social network by the transmission of the trust.

6. Experiments

In our experiments, we use a realistic social network data published on datatang [30]. The social network contains 1133 nodes and 10903 edges. The average number of neighbors of each user is 9.63. There some hub nodes in the social network. Some nodes of this social network are peers of a P2P network we construct.

A user pushes the patch to one of his friends, which is recorded as one step. We count all nodes that are patched in each step, including nodes in social network and P2P network.

The experiments are based on the assumptions as follows:

(1) A user will be immune to the corresponding vulnerability, after he has accepted the patch from a friend.
(2) The relationship between a user and his friends is one way. The trust value to a friend for a user is different from the trust value to the user for his friend.

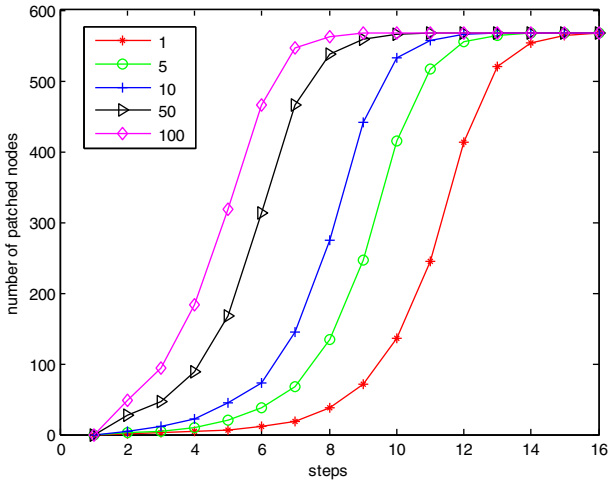

Fig. 5. The influence of the number of the initial nodes.

(3) A peer in P2P network reports the vulnerability information to the security server after being patched.

The experiment is divided into three cases as follows:

In the first case, we use the exponential patching mechanism, that is, the method in reference [5].

In the second case, we employ benign worms to patch, that is, the method in reference [21].

In the third case, we implement pushing automatic patches by social computing according to the social trust relationship.

More than 90% of nodes are users of IM network, and users who do not join the IM network generally do not use P2P network to share information.

The number of node's neighbors in social network indicates the node's degree. The higher a node's degree is, the more neighbors can help this node.

We set 50% for the vulnerability ratio, that is, 50% nodes are vulnerable, and set 1, 5, 10, 50, 100 for the number of initial nodes that the server pushed the patch to. The number of the nodes patched in the social network at each step is shown in Fig. 5, which indicates that most or all of the vulnerable nodes in the social network are patched. With the increase of the nodes the server pushed a patch to, the propagation speed of the automatic patch increases significantly.

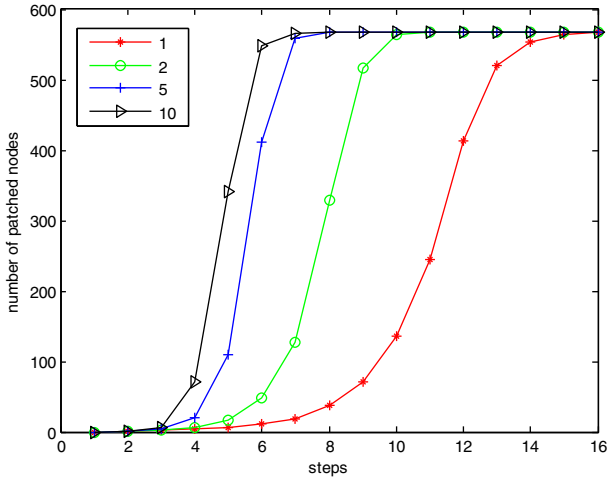

Fig. 6. The influence of the number of nodes each node pushed the patch to.

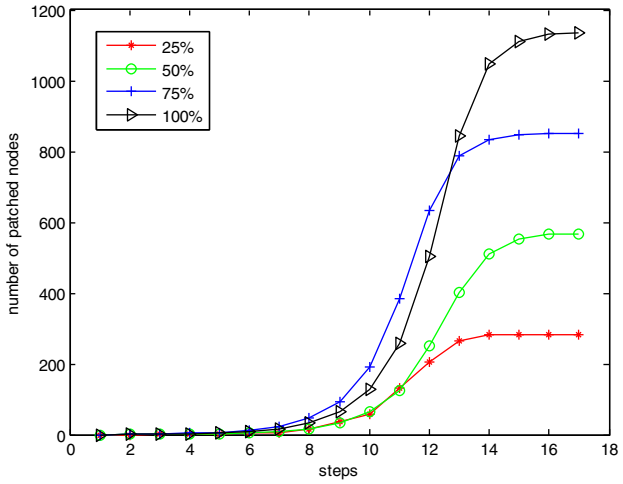

Fig. 7. The influence of the vulnerability ratio.

The number of nodes each node pushed a patch to simultaneously can influence the dissemination of the patch in social network. We set 1, 2, 5, 10 for the number of nodes which each node pushed the patch to. The number of the nodes patched in the social network at each step is shown in Fig. 6. With the increase of the nodes each node pushed a patch to, the propagation speed of the automatic patch increases significantly.

We set 25%, 50%, 75%, and 100% for the vulnerability ratio because maybe not all of the nodes have the same vulnerabilities. The node will continue to push the patch to its friends only when a node is vulnerable. The influence of different vulnerability ratio is shown in Fig. 7. With the increase of the vulnerability ratio, the propagation speed increases because the number of vulnerable nodes increases.

Some peers in P2P network have not joined any IM network, which indicates they are not in social network. We set 20%, 50%, 80%, and 100% for the coverage ratio that how many peers in P2P network are also nodes in the social network. The number of patched nodes in P2P network increases significantly with the increase of the coverage ratio, as shown in Fig. 8. In fact a user who uses P2P software also generally is a node in social network.

A node that is noticed by the request of sending the automatic patch from a friend will decide whether to receive it, according to the trust value of the friend. We set 50%, 75%, 90%, and 100% for the receiving ratio that a user decides to receive the patch. The statistical regulations are shown in Fig. 9. The propagation speed and range increases with the increase of the receiving ratio.

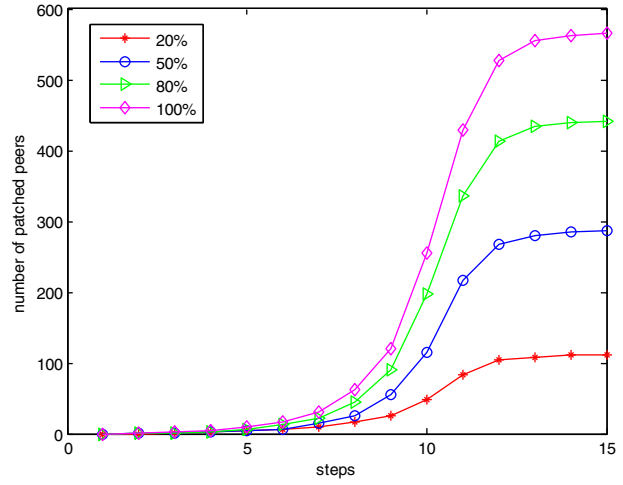

Fig. 8. The influence of the coverage ratio.

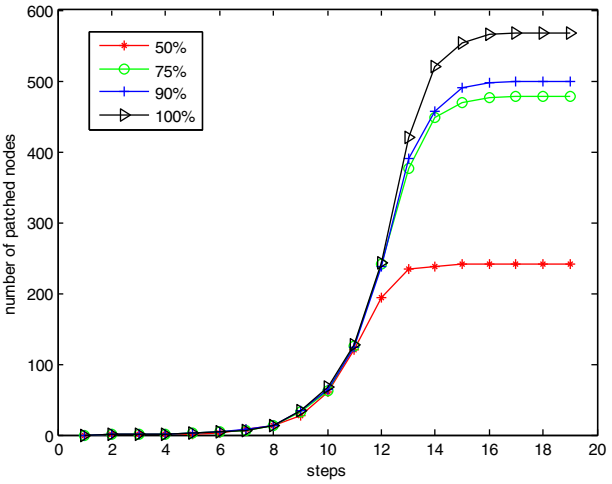

Fig. 9. The influence of the social trust.

In order to reflect the superiority of hub nodes in the patch dissemination, we set 3 cases about the order of nodes chosen by a node for pushing a patch to such as from max to min, random, and from min to max according to the node degree. The patch propagates faster through hub nodes which is the first case than other cases, as shown in Fig. 10.

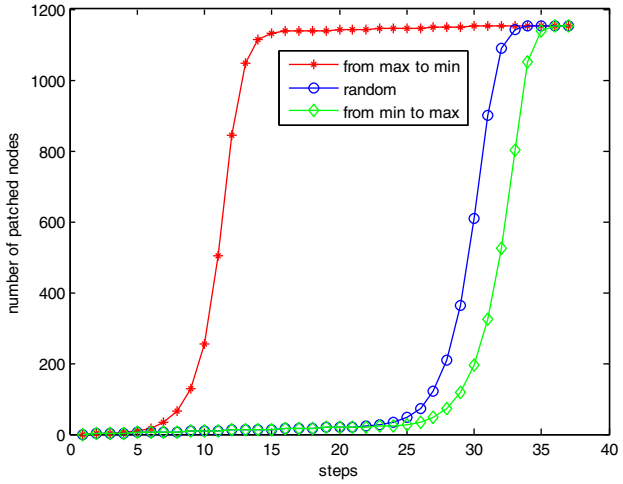

Fig. 10. The influence of the choice order.

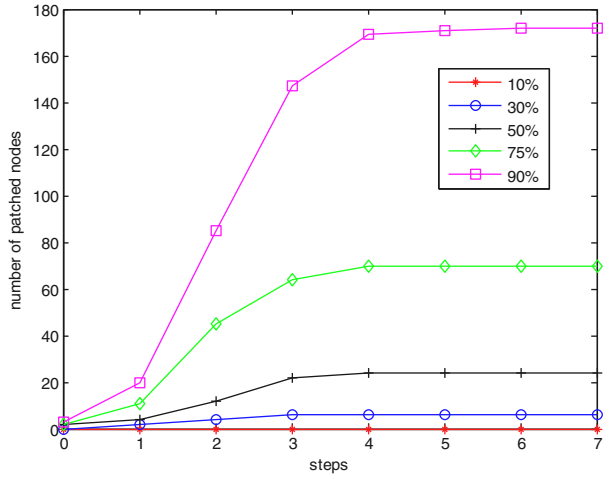

Fig. 11. The propagation of disguised patch.

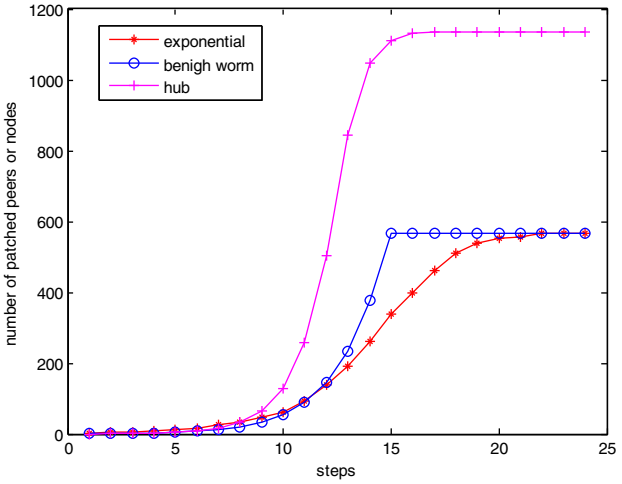

Fig. 12. The comparison of 3 mechanisms.

If a malicious code disguises an automatic patch, it will not propagate well in our system, which is shown in Fig. 11. We set the 10%, 30%, 50%, 75%, and 90% users do not identify the disguised patch and trust the sender, so they accept the patch. If 90% users accept the disguised patch, it reveals the disguised patch is likely a new malicious code because most anti-virus software cannot recognize it.

We have compared our patching mechanism with the automatic patching mechanism using benign worms and the exponential patching mechanism proposed by Shakkottai and Srikant, as shown in Fig. 12. These three methods all have implemented to patch vulnerabilities. The propagation in the automatic patching mechanism based on social computing is faster, and the propagation range is bigger than other 2 methods.

7. Conclusion

In this paper, we proposed an automatic patching mechanism based on social computing. The server can generate an automatic patch and push it to hub nodes. Those nodes accepted the patch in the social network and sent it to his friends according to the degree of his neighbors in its egocentric network by Push technology. Then the receivers accept the patch according to the trust relationship between this user and his friends. The automatic patch propagates in the social network to patch vulnerable nodes. The experimental results show that the propagation speed of the patch in the automatic patching mechanism based on social computing is faster than in the automatic patching mechanism using benign worms and in the exponential patching mechanism proposed by Shakkottai and Srikant.

Patches propagate in social network, and the security of patches is serious. For the protection of users' data, sensitive data usually have to be encrypted before outsourcing [31]. Many research focus on searching over encrypted data [32,33]. With hidden attribute-based signature technique, users are able to sign messages with any subset of their attributes issued [34]. Based on the above technology, in the future, we can improve the security of messages and files transmitted in social network.

Acknowledgement

Our works in this paper are supported by the National Natural Science Foundation of China (No. 61309024), the Natural Science Foundation of Shandong Province (Grant No. ZR2014FM038), and the project sponsored by the Scientific Research Foundation of China University of Petroleum (No. Y1207013).

References

[1] NationalComputer Virus Emergency Response Center, 2013 National Information Security, Computer and Mobile Terminal Virus Epidemic Survey Analysis Report, 2014 (Available from: http://www.antivirus-china.org.cn/diaocha2013/report2013.pdf).
[2] Microsoft Data Access Components (MDAC) Function Could Allow Code Execution(911562), Available from: http://technet.microsoft.com/zh-cn/security/bulletin/ ms06-0142006.
[3] M. Vojnovic, A. Ganesh, On the Effectiveness of Automatic Patching, Workshop on Rapid Malcode, Fairfax, Virginia, USA 2005, pp. 41–50.
[4] L. Xie, S. Zhu, A feasibility study on defending against ultra-fast topological worms, Seventh IEEE International Conference on Peer-to-Peer Computing. Galway, Ireland 2007, pp. 61–70.
[5] S. Shakkottai, R. Srikant, Peer to peer networks for defense against Internet worms, IEEE J. Sel. Areas Commun. 25 (9) (2007) 1745–1752.
[6] Friedman. "A. Good Neighbors Can Make Good Fences: A Peer-to-Peer User Security System," IEEE Technology and Society Magazine, Spring 2007, vol. 26, pp.17-24.
[7] Z. Zhu, G. Cao, S. Zhu, et al., A social network based patching scheme for worm containment in cellular networks, IEEE INFOCOM, 2009 1476–1484.
[8] S. Sidiroglou, A. Keromytis, Countering network worms through automatic patch generation, IEEE Secur. Priv. 3 (Dec. 2005) 41–49.
[9] D. Lazer, A. Pentland, L. Adamic, et al., Computational social science, Science 323 (Feb. 2009) 721–723.
[10] Golbeck J. "Weaving a Web of Trust. Science Magazine," AAAS, Sep. 2008, vol. 321, pp. 1640-1641.
[11] R. Sinha, K. Swearingen, Comparing recommendations made by online systems and friends, Proceedings of the DELOS-NSF Workshop on Personalization and Recommender Systems in Digital Libraries Dublin, Ireland, 2001.
[12] J. O'Donovan, B. Smyth, Trust in recommender systems, IUI'05: Proceedings of the 10th International Conference on Intelligent User Interfaces, New York, NY, USA, ACM 2005, pp. 167–174.
[13] J.S. Kong, B.A. Rezaei, N. Sarshar, et al., Collaborative spam filtering using e-mail networks, IEEE Comput. 39 (Aug. 2006) 67–73.
[14] M. Sirivianos, K. Kim, X. Yang, SocialFilter: Introducing Social Trust to Collaborative Spam Mitigation, IEEE INFOCOM, 2011 2300–2308.
[15] F. Zifa, J. Altmann, An empirically validated framework for limiting free-riding in P2P networks through the use of social network information, PACIS 2010. Taipei, Taiwan: Pacific Asia Conference on Information Systems, 2010.
[16] S. Marti, Trust and Reputation in Peer to Peer Networks(dissertation) Stanford University, 2005.
[17] H. Yu, M. Kaminsky, P.B. Gibbons, et al., Sybilguard: Defending against Sybil Attacks via Social Networks, Proceedings of ACM SIGCOMM 2006, pp. 576–589.
[18] Liu Xin, Jia Chunfu, Liu Guoyou, Hu Zhichao, Wang Dong, Collaborative defending scheme against malicious Web pages based on social trust, J. Commun. 33 (Dec. 2012) 11–18.
[19] J. Golbeck, Computing and Applying Trust in Web-based Social Networks(dissertation) University of Maryland, College Park, 2005.
[20] A. Abdul-Rahman, S. Hailes, Supporting trust in virtual communities, in System Sciences, Proceedings of the 33rd Annual Hawaii International Conference on. IEEE, 2000, 2000.
[21] Liu Xin Hu, Liu Guoyou Zhichao, Jia Chunfu, Defending P2P Networks Based on Benign Worms, J. Comput. Inf. Syst. 7 (2011) 2532–2539.
[22] J. Caverlee, L. Liu, S. Webb, Socialtrust: tamper-resilient trust establishment in online communities, Proceedings of the 8th ACM/IEEE-CS joint conference on Digital libraries, ACM 2008, pp. 104–114.
[23] L. Xin, S. Leyi, W. Yao, X. Zhaojun, F. Wenjing, A dynamic trust conference algorithm for social network, P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2013 Eighth International Conference on, IEEE 2013, pp. 340–346.
[24] T. Zhuo, L. Zhengding, L. Kai, Time-based dynamic trust model using ant colony algorithm, Wuhan University J. Nat. Sci. 11 (6) (2006) 1462–1466.
[25] L. Yan, Y. Pan, J. Zhang, Trust cluster head election algorithm based on ant colony systems, Computational Science and Optimization (CSO), 2010 Third International Joint Conference on, vol. 2, IEEE 2010, pp. 419–422.
[26] F.G. M´armol, G.M. P´erez, Providing trust in wireless sensor networks using a bio-inspired technique, Telecommun. Syst. 46 (2) (2011) 163–180.
[27] P. Bedi, R. Sharma, Trust based recommender system using ant colony for trust computation, Expert Syst. Appl. 39 (1) (2012) 1183–1190.
[28] Jia Chunfu, Liu Xin, Liu Guoyou, Hu Zhichao, Worm Containment Based on Double-neighbor Lists in P2P Overlay Networks, Proceedings of 2010 IEEE International Conference on Information Theory and Information Security (ICITIS 2010) December 17-19, 2010, pp. 558–562 (Beijing, China).
[29] Y. Wang, J. Lü, F. Xu, L. Zhang, A trust measurement and evolution model for internetware, J. Softw. 17 (4) (2006) 682–690.
[30] Datatang, http://www.datatang.com/.
[31] Zheli Liu, Xiaofeng Chen, Jun Yang, Chunfu Jia, Issue You, New order preserving encryption model for outsourced databases in cloud environments, J. Netw. Comput. Appl. (2014)http://dx.doi.org/10.1016/j.jnca.2014.07.001 (Elsevier).
[32] Baojiang Cui, Zheli Liu*, Lingyu Wang, Key-Aggregate Searchable Encryption (KASE) for group data sharing via cloud storage, IEEE Trans. Comput. (2015)http://dx.doi.org/10.1109/TC.2015.2389959.
[33] Jin Li, Qian Wang, Cong Wang, Ning Cao, Kui Ren, Wenjing Lou, Fuzzy Keyword Search over Encrypted Data in Cloud Computing, Proceeding of the 29th IEEE International Conference on Computer Communications (INFOCOM 2010), IEEE 2010, pp. 441–445.
[34] Jin Li, Kwangjo Kim, Hidden attribute-based signatures without anonymity revocation, Inf. Sci. 180 (9) (2010) 1681–1689 (Elsevier).