Computer Networks 57 (2013) 3522–3544

Contents lists available at ScienceDirect

Computer Networks

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

SGor: Trust graph based onion routing

Peng Zhou [a,*], Xiapu Luo [a], Ang Chen [b,1], Rocky K.C. Chang [a]

[a] The Hong Kong Polytechnic University, Hong Kong
[b] University of Pennsylvania, United States

ARTICLE INFO

Article history:
Received 27 January 2013
Received in revised form 3 July 2013
Accepted 5 August 2013
Available online 24 August 2013

Keywords:
Trust graph
Anonymity
Onion routing

ABSTRACT

The use of trust for onion routing has been proved effective in thwarting malicious onion routers. However, even state-of-the-art trust-based onion routing protocols still suffer from two key limitations in protecting anonymity. First, these protocols have no means to verify the correctness of the trust they rely on. Second, they run a high risk of being deanonymized by an inference attack due to biased trust distributions. In this paper, we propose SGor, a trust graph based onion routing that mitigates the key limitations of trust in protecting anonymity. SGor is novel with three unique properties. First, SGor aggregates group trust from mutual friends to verify the correctness of users' trust assignments. Second, SGor employs an adaptive trust propagation algorithm to derive global trust from trust graph. The global trust removes the restriction of users' local knowledge and defeats inference attacks by guiding users to discover and trust more honest routers (i.e., reducing the bias of trust distribution). Third, SGor is designed to operate in a fully decentralized manner. This decentralized design mitigates the leakage of a priori trust relationships. We evaluate SGor with simulation-based experiments using several real-world social trust datasets. The experimental results confirm that SGor can mitigate key limitations in the use of trust for protecting anonymity but introduces only a few overheads.

1. Introduction

Trust-based onion routing has recently attracted considerable attention from research communities [1–3]. Onion routing networks hide users' identities behind a circuit of selected onion routers. However, they run a high risk of being compromised in the presence of the adversaries who employ malicious onion routers to perform correlation-like attacks [4–13]. Due to a lack of effective trust models, it is very difficult for users to evade adversaries' routers when establishing onion circuits. As a result, the adversaries who control a large fraction of onion routers pose a serious threat to existing onion routing networks (e.g., the Tor network [14]). To thwart the correlation-like attacks, recent research proposes trust-based onion routing. With the trust that users have readily assigned to routers' owners, adversaries' routers can be identified and excluded from users' onion circuits.

However, existing trust-based onion routing computes trust only according to users' own knowledge [2,3]. This trust is local and subject to two key limitations. First, since people could have inaccurate knowledge about others, users' local trust on routers' owners could be incorrect [3]. If a user wrongly trusts an adversary, trust-based onion routing has no effective ways to remove this incorrect trust assignment. Second, based on users' own knowledge, users can trust only the onion routers deployed by their acquaintances. Users are forced to blindly treat their unfamiliar routers as adversaries. Since normal users usually have knowledge only for a small group of onion routers and different users have knowledge for different fractions of the network, the resulting trust distribution could be largely biased. This biased distribution

* Corresponding author. Tel.: +852 62134272.
E-mail addresses: cspzhouroc@comp.polyu.edu.hk (P. Zhou), csxluo@comp.polyu.edu.hk (X. Luo), angchen@cis.upenn.edu (A. Chen), csrchang@comp.polyu.edu.hk (R.K.C. Chang).
[1] This work was done when the co-author was with The Hong Kong Polytechnic University.

1389-1286/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.comnet.2013.08.005

leads honest routers to be trusted by a small number of users. For this reason, an adversary who knows a priori trust relationships has a high probability to guess the initial user of an onion circuit if the adversary can observe a router in this circuit (i.e., inference attack). As an extreme example, if an adversary Pete observes a Bob's router in an onion circuit and has a priori knowledge that Bob is only trusted by Alice, Pete can therefore immediately confirm that the initial user of this onion circuit is Alice.

To effectively protect anonymity using trust in onion routing networks, a fundamental challenge is to eliminate aforementioned limitations. However, no complete solution has been proposed yet to address this challenge in previous studies. To the best of our knowledge, no prior research has provided solutions that enhance the correctness of trust assignments in trust-based onion routing. Moreover, although several inference attack countermeasures have been proposed, the root cause of this attack has not been addressed. For example, to thwart inference attacks, a downhill algorithm [3] employs a decreasing trust threshold along onion circuits, while a trust degree based algorithm [15] takes other users' trust into consideration. These counters cannot reduce the bias of a priori trust distributions, hence missing the opportunity to solve the root cause of inference attacks.

In this paper, we propose SGor, a novel trust graph based onion routing that mitigates key limitations in the use of trust for protecting anonymity. SGor is designed based on two key insights. First, if people can assign trust to others according to their own knowledge independently, the trust from a group of honest people is more likely to be correct than the trust from a single honest person. Based on this observation, SGor aggregates group trust from mutual friends. The group trust enables users to verify the correctness of their trust assignments by consulting their friends' independent opinions. Second, although users have no immediate knowledge for their unfamiliar routers, these routers are not necessarily controlled by adversaries. This observation motivates us to propose an *adaptive trust propagation* algorithm. By guiding users to discover more honest routers even if they have no immediate knowledge about these routers, the proposed algorithm can derive global trust from a trust graph. The global trust leads honest routers to be trusted and selected by more users and thus impedes inference attacks by alleviating the bias of trust distributions.

Moreover, SGor is designed to run in a fully decentralized manner. This design benefits SGor in two aspects. First, our decentralized algorithms do not require users to expose their local trust relationships to other third parties (e.g., a centralized server). Since inference attacks can only be successfully launched with the knowledge of a priori trust relationships, this design assists SGor to evade inference attacks by mitigating the leakage of a priori trust relationships. Second, the removal of a centralized server can help SGor adapt efficiently to a large-scale trust graph with low costs.

To sum up, our contributions in this paper are threefold:

1. We propose SGor, a new onion routing protocol that protects anonymity using a trust graph.
2. We design novel decentralized algorithms to derive group trust and global trust from the trust graph. The group trust can be used to enhance the correctness of trust assignments, and the global trust is effective in reducing the bias of trust distributions. Using these new trust features, SGor mitigates key limitations in the use of trust for protecting anonymity.
3. We evaluate SGor with extensive simulation-based experiments using real-world social trust datasets. The experimental results confirm that SGor can make an effective use of the trust graph to protect anonymity but introduce only a few overheads.

The remainder of this paper is organized as follows. We start by revisiting state-of-the-art trust-based onion routing in Section 2. We present a high level overview of SGor in Section 3. We elaborate on the design of SGor in Section 4. We evaluate SGor using real-world social trust datasets in Section 5. After discussing several limitations in Section 6, we conclude SGor in Section 7.

2. State-of-the-art

Since users always expect to prevent their identities (usually in the form of IP addresses) from leaking to others when they visit the Internet, anonymous communication becomes an essential part to Internet communications. Onion routing protocol [14,16] is one of the most dominated anonymous communication techniques in the Internet today. It wraps users' traffic using successive layers of encryption along a circuit of selected onion routers, hence hiding the initial users behind the onion circuits. However, if users select adversaries' onion routers in their circuits, they are more likely to be deanonymized by various correlation-like attacks [4–13]. For example, adversaries can passively analyze traffic characteristics and patterns [4–9,12] or actively embed traffic watermarks [4,10,11,13] to correlate the onion routers controlled by them, hence largely reducing the anonymity protected by onion routing networks. Due to the lack of an effective mechanism that verifies the identities and technical competence of routers' owners, the correlation-like attack is hard to be prevented by existing onion routing networks.

To defeat correlation-like attacks, recent research proposes the use of trust for excluding adversaries' routers. For example, Krishna et al. [1] restrict users to only select onion routers from their 1- and 2-hop friends in an online social network. Drac [17] and Pisces [18] perform random walks through social links on top of an online social network to discover honest routers. Each directed social link represents a trust that one person assigns to another person. However, these three studies cannot give an in-depth analysis for discussing adversary models and optimal trust-based routing algorithms. To fill in this gap, Johnson et al. provide a general model for trust-based onion routing [2]. This model includes completely theoretical adversary models and discusses corresponding optimized trust-based onion routing algorithms in the presence of these adversary models. Moreover, Johnson et al. design an enhanced model to analyze trust-based onion routing in practical scenarios [3]. This practical model considers that different users have different distributions of local trust among onion routers because users are usually attacked by different adversaries in the wild. This practical model also studies how the accuracy of trust assignments affects the effectiveness of trust-based onion routing.

Although these trust-based onion routing protocols [1–3,17,18] employ trust to successfully thwart correlation-like attacks, they still suffer from two key problems that limit the effectiveness of protecting anonymity.

2.1. Incorrect trust

In existing trust-based onion routing, users compute trust only according to their own knowledge about the owners of onion routers. This trust could be incorrect [3] because normal users usually have limited competence of censoring others. Incorrect trust assignments could induce adversaries' routers into users' onion circuits and thus limit the effectiveness of protecting anonymity in trust-based onion routing.

Pisces [18] proposes a tit-for-tat algorithm to prevent random walks from going through incorrect social links. Using this algorithm, each honest router works as a runtime watch dog and monitors other routers listed in its routing table. If an honest router A detects that another router B has excluded A from B's routing table, A could consider that B is an adversary and remove B from A's routing table as a revenge. However, this algorithm needs a heavy cost in runtime, and could be easily avoided because it does not utilize additional trust. For example, instead of removing honest routers from routing tables, smart adversaries can simply use a very low probability to choose honest routers in random walks.

Unlike Pisces [18], SGor employs a new trust feature, called group trust, to verify the correctness of trust assignments. The group trust cannot be easily avoided because it vouches the correctness of trust assignments using additional knowledge from honest people.

2.2. Biased trust distribution

Since users assign trust according to their own knowledge and different users usually have knowledge only for different and small fractions of the network, the resulted trust distributions are more likely to be biased and honest routers are usually trusted by only a small set of users. For this reason, the adversaries who have the knowledge of a priori trust relationships have chances to guess the initial users of onion circuits if they can observe routers in these circuits (i.e., inference attacks).

Prior research has proposed two countermeasures to circumvent inference attacks. One is a downhill algorithm that enables users to select routers along their onion circuits from sets with a decreasing trust threshold [3]. The other is a trust degree based solution that optimizes trust-based router selections by considering other users' trust assignments [15]. However, since these two countermeasures cannot reduce the bias of trust distributions, they only provide restricted capabilities of evading inference attacks.

To reduce the bias of trust distributions, SGor proposes the use of global trust from a trust graph to thwart inference attacks. Although SGor is not the first solution that applies global trust to onion routing, it is novel because it can mitigate the chance of introducing adversaries when deriving global trust. Unlike Drac [17] and Pisces [18] which perform conventional or metropolis–hastings random walks through social links to implicitly utilize global trust, SGor designs a novel adaptive trust propagation algorithm that computes global trust by using group trust to rate limit trust propagation. This algorithm mitigates the probability that global trust is propagated through incorrect trust edges, hence preventing adversaries from compromising global trust during trust propagation.

3. SGor overview

In this section, we describe SGor in a high level. We first set design goals for SGor in Section 3.1. We then list basic assumptions in Section 3.2. After elaborating on threat model and trust model in Sections 3.3 and 3.4 respectively, we present an overview of SGor with its architecture and major components in Section 3.5.

3.1. Design goals

To effectively protect anonymity using trust, SGor is expected to meet the following four key requirements.

Trust Graph: To evade malicious routers and thwart correlation-like attacks, SGor is required to have the capability of constructing onion circuits using trust from a trust graph.
Group Trust: To verify the correctness of trust assignments, SGor is required to have the capability of aggregating group trust from mutual friends.
Global Trust: To reduce the bias of trust distributions and hence defeat inference attacks, SGor is required to have the capability of deriving global trust from a trust graph.
Decentralization: To prevent users' trust relationships from leaking to any third parties and adapt efficiently to a large scale trust graph, SGor is required to have a fully decentralized architecture.

These requirements work together to distinguish SGor from past works in the literature [1–3,17,18].

3.2. Basic assumptions

We make three basic assumptions for the design of SGor.

1. Users and routers' owners in an onion routing network are also members of a trust network.
2. Users and routers' owners can assign trust to others according to their own knowledge independently.

3. Users and routers' owners can run user-agents in their own machines (e.g., in-browser add-ons, desktop or mobile phone installs) to support SGor's decentralized router selection process in a fully automatic manner.

Assumption 1 defines the potential population who can use SGor. For example, if we derive a trust graph from an online social network and apply it to SGor, the members of this online social network can use SGor to protect their anonymity when they visit Internet services. As reported by SocialBakers (www.socialbakers.com) in July 2012, Facebook, the most popular online social network, has more than 870 million members. Moreover, there are more than 2.2 billion Internet users in the world according to a report from InternetWorldStats (www.internetworldstats.com) in December 2011. Apparently, if SGor adopts Facebook as its trust graph, at least 40% Internet users who have accounts in Facebook can use SGor when they access Internet through onion routing.

With Assumption 2, SGor is built upon a trust graph with independent trust assignments. If users can assign trust to others independently, they can avoid the influence of incorrect trust over a group of people. Otherwise, the group trust could have weak capability of verifying the correctness of trust assignments. For example, if Bob assigns trust to Pete due to the reason that Bob's friend Alice has already trusted Pete (i.e., Bob's trust on Pete is correlated with Alice), Alice cannot rely on the group trust from Bob to verify the correctness of her trust in Pete, because Bob's trust is inherited from Alice. In real world, the friendship graph of an online social network is more likely to have correlated trust assignments, because people usually make friends by consulting their old friends' friendship circles. Unlike that, the interaction graph [19,20] admits trust assignments if and only if users have direct communications, hence being more independent. For this reason, we evaluate SGor by adopting the interaction graph in Section 5.

With Assumption 3, users can select onion routers from others and onion routers' owners can provide onion routers to others in a fully decentralized and automatic manner. For example, if SGor adopts the interaction graph of Facebook as its trust graph, we can develop a Facebook App (http://developers.facebook.com) for SGor. All the users and routers' owners run this App in their own machines. The running Apps communicate with each others in the Internet to accomplish the decentralized router selection process automatically. The communication traffic is encrypted to prevent potential leakage.

3.3. Threat model

To compromise users' anonymity protected by (trust-based) onion routing networks, we consider adversaries have the following attacking capabilities:

3.3.1. The capability of compromising onion routers

First, we assume adversaries can arbitrarily deploy their own malicious routers to existing onion routing networks. This assumption holds in real-world onion routing networks such as Tor [14], because these networks are open to accept any volunteer's routers without checking their identities. Although some networks have runtime monitoring systems to exclude malicious routers, these systems are usually based on the uptime of normally behaving. As a result, smart adversaries can easily bypass these monitoring systems by operating their routers in normal behaviors for a long period. Second, we assume adversaries can exploit hacking techniques to compromise some honest routers which contain security vulnerabilities. This assumption adapts to advanced adversaries who master powerful computer network skills. Third, we simply assume that Internet services visited by onion routing users are already controlled by adversaries. This assumption allows us to design SGor by considering the worst scenario, because users can visit any Internet servers through onion routing and it is nearly impossible to confirm which servers are compromised and which are not.

3.3.2. The capability of locally observing onion routers

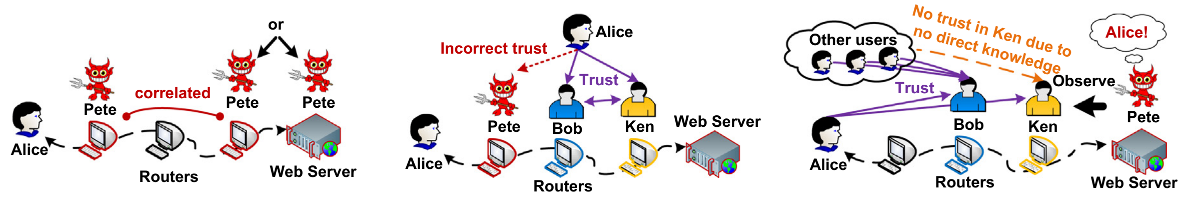
We assume that adversaries can observe onion routers in users' onion circuits. This capability is required by the adversaries who perform inference attacks. However, since existing onion routing protocols are not designed to resist global adversaries who can monitor the whole communication infrastructure of onion routing networks (e.g., ISP-level or state-level adversaries) [14,16], we also assume that adversaries can only employ compromised onion routers to observe adjacent routers in onion circuits or use compromised Internet servers to observe the last router in onion circuits.

3.3.3. The capability of correlating malicious onion routers

We assume that adversaries can correlate the onion routers and Internet servers under their control in the same onion circuit (i.e., the capability of performing correlation-like attacks). This assumption holds in real-world attacking scenarios, because several prior studies have demonstrated that various traffic watermarking and analysis techniques are effective in correlating two compromised endpoints (e.g., onion routers or Internet servers) in the same onion circuit [9,11,12,21–24]. Fig. 1(a) illustrates an example of the correlation-like attack against onion routing networks. In this example, Alice is a user who has no effective method to verify router identities. Pete is an adversary who control onion routers. If Alice selects Pete's routers as the first router in her onion circuit and visits an Internet server under Pete's control, Pete can correlate Alice and the server Alice is visiting.

3.3.4. The capability of attracting incorrect trust

We assume that adversaries can exploit users' inaccurate knowledge to attract incorrect trust assignments. Since trust-based onion routing computes trust according to users' own knowledge about routers' owners, if users mistakenly trust an adversary, they could trust all the routers controlled by this adversary. Fig. 1(b) shows an example of an incorrect trust assignment in trust-based onion routing. If Alice makes a mistake to trust an adversary Pete, she could select Pete's routers in her onion circuit. Moreover, although we agree that adversaries can easily deploy a large number of fake accounts in trust graph (i.e., Sybil



(a) Threat model for correlation-like attack. (b) Threat model for incorrect trust. (c) Threat model for inference attack.

Fig. 1. Threat models.

attack [25]), it is very difficult for them to gain incorrect trust from a large number of honest humans. Previous social network based Sybil defences also adopt this assumption because this assumption widely holds in real-world social networks [26].

3.3.5. The capability of collecting a priori trust relationships

We assume that adversaries can collect a priori trust relationships of a trust graph. For example, if we apply an online social network as a trust graph to SGor, adversaries could employ a crawler to run a brute-force scan of this social network. This approach can reveal the local trust relationships of users who make available their private information to the public [27,28]. To obtain the trust relationships of users who could only expose their private information to their friends, adversaries could employ socialbots [29] to make friends with these users, or estimate these users' relationships using the leakage of their friends' trust relationships (i.e., link privacy leakage [30]). Using a priori trust relationships, adversaries can perform inference attacks and hence largely reduce users' anonymity protected by trust-based onion routing. Fig. 1(c) demonstrates an example of inference attack against trust-based onion routing. In this demonstration, Pete is an adversary who knows a priori trust relationships. If Pete observes Ken's routers in an onion circuit, he can guess Alice as the initial user of this circuit because Ken is only trusted by Alice. However, Ken's routers are in fact honest. Other users do not trust Ken just because they have no knowledge about Ken. It is not due to that they confirm Ken is an adversary.

3.4. Trust model

We model SGor's trust graph as a directed graph $G = (V, E)$. V is the set of nodes in G. A node $v_i \in V$ represents a person in the trust graph. E is the set of edges in G. An edge $v_i \rightarrow v_j = e_{ij} \in E$ indicates that the person $v_i \in V$ assign local trust to another person $v_j \in V$ in the trust graph (i.e., a trust edge represents a local trust assignment). $|V|$ and $|E|$ are the numbers of nodes and edges in G, respectively. As discussed in [2,3], users can only assign coarse level trust to others. SGor therefore only considers two levels of local trust assignments, *trust* and *distrust*. If $v_i \rightarrow v_j$ exists, v_i trusts v_j. Otherwise, v_i distrusts v_j. Moreover, we consider the trust is positively transitive [31]. That means a friend of Alice's friend is still a friend of Alice.

We can sample SGor's trust graph G using any real-world trust networks. For example, we can derive a social trust graph from an online social network or adopt a reputation-based trust graph from an online community. We can also design an open community that facilitates users and routers' owners to evaluate the trustworthiness of each other independently, hence generating a customized trust graph to support SGor. In this paper, we mainly focus on the design of SGor using online social networks because this choice makes SGor easy to be applied in the Internet.

In the literature, there are two models that can be used to sample the trust graph G from an online social network. One is the friendship model and the other is the interaction model [19,20]. They have different capability to express the trust that already exists in online social networks. We take an example from Facebook to show the difference. We first assume Alice and Ken are two persons with accounts in Facebook. If Alice attempts to make a friend with Ken, she sends a *friend request* to Ken. If Ken confirms this request, they are shown in each other's friendship list. The friendship model treats this kind of relationship as trust. The interaction model is built on top of the friendship model. If Alice has local trust with Ken in an interaction model, only being a friend of Ken is not enough. Alice also needs bi-directional communications with Ken through online social networks (e.g., posting messages with each other in walls or tagging each other in photos).

SGor employs the interaction graph to sample G because of two reasons. First, the local trust in the interaction graph is more independent. A person may be easy to make a new friend who is already a friend of his old friends, but they are less likely to have bi-directional communications with each other if they are not acquaintances. Second, recent research has advocated that interaction graphs can better represent real world trust than friendship graphs [19,20].

For the ease of description in the following sections, although we do not use the friendship graph to sample G, we also regard v_j as v_i's friend if v_i assigns local trust to v_j (i.e., $v_i \rightarrow v_j$ exists).

3.5. SGor architecture and major components

In this section, we gives a high level overview to introduce SGor's architecture and major components.

3.5.1. SGor architecture

SGor provides trust graph based onion routing by "overlaying" a trust network on top of the onion routing infrastructure. SGor has a two layered hierarchical architecture. The upper layer is for trust graph based router

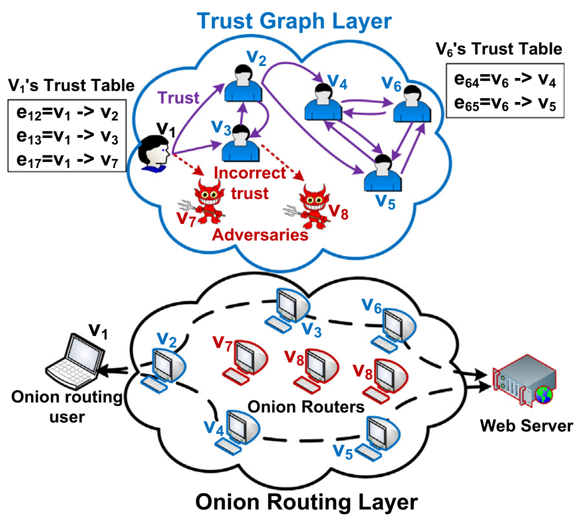


Fig. 2. Two layered hierarchical architecture of SGor.

selection, and the underlayer is for onion routing. To communicate with remote Internet servers through SGor, users should first select onion routers according to trust in the trust graph layer. They then use the selected routers to establish onion circuits in the onion routing layer. We note that SGor is novel due to its trust graph layer. The onion routing layer runs the same protocol as traditional onion routing networks.

In SGor, each person in the trust network layer can arbitrarily play any roles in the onion routing layer. In particular, a person can act as onion routing user who requires onion routers from others. This person can also operate as a router owner who provides onion routers to others. Moreover, a person can work as both of an onion routing user and a router owner concurrently, or play neither of the two roles in the onion routing layer. In the last case, the person merely assists other people to propagate trust over the trust graph.

Fig. 2 illustrates an example of the architecture of SGor. In this example, a person v_1 attempts to visit a (sensitive) Internet server through SGor. People v_2–v_6 provide onion routers in SGor. v_7 and v_8 are two adversaries. In the trust graph layer, although v_1 makes a mistake to trust the adversary v_7, SGor has a chance to exclude v_7 because no other v_1's friends trust v_7 (i.e., no other ways can be used to verify the correctness of $v_1 \rightarrow v_7$). Moreover, although v_1 only has local trust in v_2 and v_3, SGor can propagate v_1's trust to other persons v_4-v_6 in the trust graph. Through v_2, v_1's trust can be propagated to v_4 and v_5, and further to v_6. However, for security concern, v_3 cannot propagate v_1's trust to v_8 because no means can be used to verify the correctness of the local trust $v_3 \rightarrow v_8$. In the onion routing layer, v_1 constructs onion circuits using onion routers derived from the trust graph layer. Since SGor calculates trust features in the trust graph layer, onion routing users can evade all the onion routers deployed by the person whom they do not trust (e.g., v_8 has two routers in Fig. 2. If v_1 does not trust v_8, he can circumvent the two routers deployed by v_8).

3.5.2. Major components

SGor consists of three major components in its trust graph layer: trust management, trust aggregation and trust propagation. They work together to meet aforementioned design requirements for SGor (Refer to Section 3.1).

3.5.2.1. Trust management. This component provides functionalities for managing local trust relationships. Since SGor is designed to run without any central servers, SGor users have to manage local trust relationships for themselves. As shown in Fig. 2, each SGor user has a trust table to store local trust for their friends. This design results in two advantages. First, this design does not require any people to expose their local trust to any others, hence mitigating the leakage of a priori trust relationships. Second, this design adapts SGor to a large scale trust graph with a low cost in the storage. Rather than all the trust relationships in the entire trust graph, each SGor user is only required to store trust relationships for their friends.

3.5.2.2. Trust aggregation. This component aggregates group trust from mutual friends, hence offering countermeasures to remove incorrect trust assignments. A person's friend receives group trust when this friend is also trusted by many other friends. The group trust can be used to remove incorrect trust assignments, because people can verify the correctness of their local trust assignments by consulting other friends' independent opinions. This component has a basic function to aggregate group trust from mutual friends. Moreover, since people could join and leave the trust network dynamically, and update their local trust assignments when their knowledge is renewed, this component also provides an additional function to handle this dynamic behavior.

3.5.2.3. Trust propagation. This component computes global trust by propagating local trust though trust graph, hence leading honest routers to be trusted by more users. This component plays the key role in reducing the bias of trust distribution and hence resisting inference attacks. We implement an *adaptive trust propagation* algorithm for this component. Using this algorithm, a person can propagate trust from the people who trust him to the other people whom he trusts. For example, Alice trusts Bob and Bob trusts Ken. If Bob propagates the trust from Alice to Ken, Alice could also trust Ken. Moreover, to mitigate the risk of mistakenly propagating trust to adversaries, SGor takes a counter that adapts trust propagation capacity to group trust (i.e., people use group trust to limit the maximum number of people to whom they can propagate trust through a friend). Back to the foregoing example, if Alice has a higher level group trust in Bob, she can propagate trust through Bob to more other people. But if Alice has a lower level group trust in Bob, she could just propagate trust through Bob to fewer, or even none, other people. By adaptively propagating trust over the entire trust graph, people can discover more honest routers and thus mitigate the bias of trust distributions. In return, these honest routers can be trusted and selected by more users and become more effective in thwarting inference attacks.

4. SGor design

In this section, we elaborate on the design of SGor. First, we apply an algorithm to aggregate group trust from mutual friends in Section 4.1. Second, we propose an adaptive trust propagation algorithm to derive global trust from trust graph in Section 4.2. Based on these two new trust features, we design a trust graph based router selection algorithm in Section 4.3. All our proposed algorithms run in a fully decentralized manner. We also analyze SGor's capability of protecting anonymity using a probabilistic model in Section 4.4.

For the ease of reference, we summarize important notations used by this paper in Table 1.

4.1. Group trust

SGor employs group trust to verify the correctness of local trust assignments. In this section, we discuss the details for aggregating group trust. We first describe the concept of robust trust path by analyzing security concerns of trust path. We then give the definition of group trust based on robust trust path. Afterwards, we design a decentralized algorithm to aggregate group trust on top of a trust graph.

4.1.1. Robust trust path

In trust graph $G = (V, E)$, we consider v_i has a trust path to v_j if v_i can reach v_j through a sequence of successive trust edges. A trust path from v_i to v_j implicitly indicates that v_i trusts v_j, hence providing a unique way for v_i to confirm that v_j is not an adversary (the local trust $v_i \rightarrow v_j$ is correct).

We consider that a trust path is robust if this path cannot be arbitrarily forged through a single incorrect trust edge. However, not all the trust paths are necessarily robust. For example, if v_i assigns incorrect local trust to two adversaries v_j and v_k (i.e., $v_i \rightarrow v_j$ and $v_i \rightarrow v_k$ exist in G), the adversary v_k can forge unlimited number of trust paths from v_i to v_j because adversaries can arbitrarily trust other adversaries. Fig. 3 demonstrates forged trust paths. v_1 has assigned an incorrect local trust to v_2 (i.e., $v_1 \rightarrow v_2$ is incorrect). If v_3 is also an adversary (i.e., $v_1 \rightarrow v_3$ is incorrect), he can arbitrarily forge trust paths from v_1 to v_2. As shown in this figure, the adversary v_3 has forged 2, 3 and 4 trust paths from v_1 to v_2.

In this paper, we find that the trust paths consisting of no more than two trust edges are robust. We give an

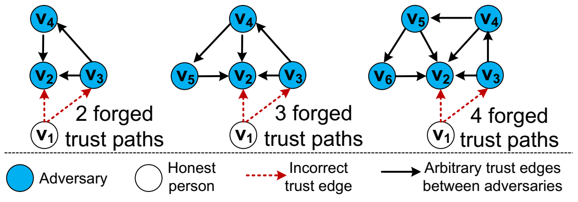


Fig. 3. Examples of forged trust paths.

example to explain this finding. Considering two trust paths from v_i to v_j, $v_i \rightarrow v_k \rightarrow v_j$ is a robust trust path while $v_i \rightarrow v_k \rightarrow v_m \rightarrow v_j$ is not. The reason is that, if v_k is an adversary, he can arbitrarily trust any other adversaries like v_m, hence forging an unlimited number of trust paths like $v_i \rightarrow v_k \rightarrow v_m \rightarrow v_j$. However, the adversary v_k can only forge one robust trust path $v_i \rightarrow v_k \rightarrow v_j$ from v_i to v_j.

Since robust trust paths cannot be arbitrarily forged by adversaries (i.e., one incorrect trust edge can only corrupt one robust trust path), they can provide robust ways to confirm that a local trust assignment is correct.

4.1.2. Group trust definition

Let Φ_{ij} be the *group trust* v_i has in v_j. Φ_{ij} can be calculated by counting up the number of robust trust paths from v_i to v_j. We have $\Phi_{ij} = 0$ if the trust edge $v_i \rightarrow v_j$ does not exist.

If v_i is an honest person but v_j is an adversary, Φ_{ij} reflects the number of incorrect trust edges accompanied with the incorrect trust edge $v_i \rightarrow v_j$. Theorem 1 proves this nature.

Theorem 1. *If v_i is an honest person, v_j is an adversary and $\Phi_{ij} = N$, there must exist N incorrect trust edges in the robust trust paths from v_i to v_j.*

Proof. We use mathematical induction for the proof.

Base case: Consider $\Phi_{ij} = 1$, v_i can only have one robust trust path to v_j, i.e., $v_i \rightarrow v_j$. Meanwhile, this robust trust path consists of only one trust edge $v_i \rightarrow v_j$. As a result, if v_j is an adversary, we have 1 incorrect trust edge $v_i \rightarrow v_j$.

Inductive step: Assuming Theorem 1 holds for $\Phi_{ij} = N$, we show that Theorem 1 also holds for $\Phi_{ij} = N + 1$. Compared with $\Phi_{ij} = N$, v_i has an additional robust trust path to the adversary v_j when $\Phi_{ij} = N + 1$. We can simply assume

Table 1
Important notations.

Notation	Definition	Notation	Definition
G	A trust graph. $G = (V, E)$	V, E	V is a set of nodes in G, E is a set of edges in G
Φ_{ij}	The *group trust* that a person v_i has in another person v_j	Φ_h	A minimum group trust threshold
t_{ij}	A random token the person v_i generates and sends to his friend v_j	C_{ij}	The number of routers to which v_i can propagate trust through the friend v_j
L_i	The maximum number of hops to which v_i can propagate trust	R_i	The number of routers a person v_i expects to collect through trust propagation
T_{ij}	The number of v_i's tickets that the person v_j receives	$P(A_j \mid \Phi_{ij})$	The probability that v_j is adversary given v_i has group trust Φ_{ij} for v_j
r_j	The number of routers that the person v_j deploys in SGor	$P(A_j \mid L_i)$	The probability that v_j is adversary given v_i propagates trust to v_j through L_i hops
λ	$P(\Phi_{ij} \mid A_j)$ follows a Poisson distribution with a parameter λ	β	$P(A_j)$ is independent and identically distributed with a parameter $\beta \in (0, 1)$

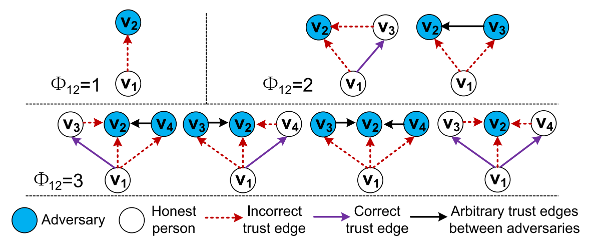


Fig. 4. If v_1 is an honest person but v_2 is an adversary, the group trust Φ_{12} equals to the number of incorrect trust edges in the robust trust paths from v_1 to v_2.

this additional path is $v_i \rightarrow v_k \rightarrow v_j$. We consider two cases for this path: (i) v_k is an adversary and (ii) v_k is an honest person. For case (i), $v_i \rightarrow v_k$ is an additional incorrect trust edge. While for case (ii), $v_k \rightarrow v_j$ is incorrect because v_j is an adversary. As a result, if $\Phi_{ij} = N$ indicates N incorrect trust edges in the robust trust paths from v_i to the adversary v_j, $\Phi_{ij} = N + 1$ can lead to $N + 1$ incorrect trust edges. □

Fig. 4 gives examples for Theorem 1. It can be seen that if $\Phi_{12} = 1$ and v_2 is an adversary, we have only 1 incorrect trust edge, i.e., $v_1 \rightarrow v_2$. However, if $\Phi_{12} = 2$, 2 incorrect trust edges are accompanied. In particular, $v_1 \rightarrow v_2$ and $v_3 \rightarrow v_2$ are incorrect if v_3 is an honest person, or $v_1 \rightarrow v_2$ and $v_1 \rightarrow v_3$ are incorrect if v_3 is an adversary. Similarly, $\Phi_{12} = 3$ confirms that there are 3 incorrect trust edges.

Since group trust Φ_{ij} is the total number of robust trust paths from v_i to v_j, a larger Φ_{ij} indicates that v_i can obtain more robust means to confirm that v_j is not an adversary. SGor therefore use Φ_{ij} to validate the correctness of $v_i \rightarrow v_j$. In the design of SGor, users can set a minimum group trust threshold Φ_h, and have more confidence to confirm that $v_i \rightarrow v_j$ is correct (i.e., v_j is not an adversary) if $\Phi_{ij} \geqslant \Phi_h$.

4.1.3. Group trust aggregation algorithm

SGor applies a decentralized algorithm to aggregate the group trust Φ_{ij} by counting the number of v_i's friends who have v_j as a mutual friend of v_i. This algorithm is based on two observations. First, if v_i has a robust trust path $v_i \rightarrow v_k \rightarrow v_j$ through v_k to v_j, v_j is a mutual friend of v_i and v_k. Second, the group trust Φ_{ij} is the total number of robust trust paths from v_i to v_j.

This group trust aggregation algorithm runs three steps in a fully decentralized manner. The communications between any two people in the trust graph layer are assumed to be encrypted.

Step 1: At first, an initial person generates random tokens on the fly, and maps different tokens to different friends. The initial person sends each token to this token's mapped friend. Since every token is marked by the initial person, the receivers can confirm these tokens are initial tokens because they are sent and marked by the same person. As shown in Fig. 5, v_1 has trust edges to v_2, v_3, v_4 and v_5. To aggregate group trust for these friends, v_1 generates and maps random tokens t_{12}, t_{13}, t_{14} and t_{15} to v_2, v_3, v_4 and v_5, respectively. v_1 then sends these initial tokens to their mapped friends (i.e., sending t_{1x} to v_x, where $x = 2, 3, 4, 5$).

Step 2: When a person receives an initial token, he will further forward this token to his friends. These forwarded tokens are sent and marked by different people. Receivers regard them as second hand tokens. As shown in Fig. 5, v_2 forwards t_{12} to v_3, while v_3 and v_5 forward t_{13} and t_{15} to v_4. Actually, v_2, v_3, v_4 and v_5 should forward the tokens received from v_1 to all of their friends, even if these friends are not v_1's friends. We omit this process in Fig. 5 to make a clear demonstration.

Step 3: When a person receives second hand tokens, he can take actions depending on whether he has already received an initial token with the same marker of the second hand tokens. If it is not the case, this person will discard these second hand tokens. Otherwise, he will send the initial token and all the second hand tokens back to the initial person who marks them. The initial person calculates the group trust for each of his friends by counting the number of tokens returned from these friends. In Fig. 5, v_2, v_3, v_4 and v_5 return 1, 2, 3 and 1 tokens to v_1, respectively. Hence, v_1 obtains $\Phi_{12} = 1$, $\Phi_{12} = 2$, $\Phi_{13} = 3$ and $\Phi_{14} = 1$.

Since the proposed group trust aggregation algorithm operates in a fully decentralized manner, there are three scenarios that a person has to start running this algorithm: (i) a fresh person first joins SGor; (ii) a person changes local trust assignments to his friends; (iii) a person has friends who change their local trust assignments to these friends' friends.

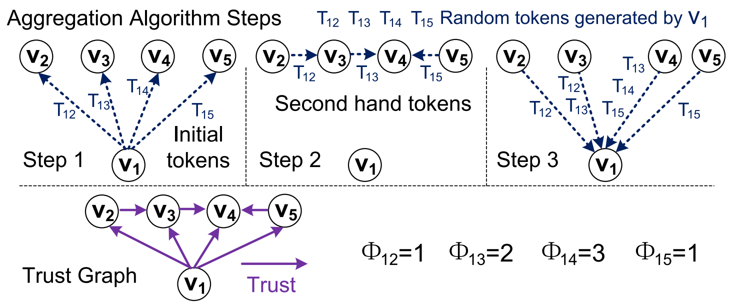


Fig. 5. The steps of group trust aggregation algorithm.

4.2. Global trust

SGor employs global trust from trust graph to thwart inference attacks. We consider a person v_i has global trust in another person v_j if v_i has at least one trust path to v_j. Using global trust, onion routing users can trust routers even if they have no direct knowledge for these routers' owners.

SGor derives global trust through trust propagation. Considering people v_i and v_j, v_i has global trust in v_j if and only if v_i can propagate his trust to v_j through a trust path. However, if people naively propagate trust through any trust paths without any constraints, they cannot limit the number of adversaries who receive global trust during trust propagation. For example, if v_k can propagate v_i's trust but v_k is an adversary, v_k can arbitrarily propagate v_i's trust to any other adversaries.

To address this problem, we propose an adaptive trust propagation algorithm. The key idea is to limit the number of people to whom a person can propagate trust through a friend. The limitation is determined by the group trust that this person has in this friend. This algorithm defends trust propagation against adversaries in two aspects. First, even if an honest person mistakenly propagates trust to an adversary, this adversary can only further propagate trust to a limited number of other adversaries. Second, since a larger group trust received by a friend indicates a lower probability that this friend is an adversary, the friend who is less likely to be an adversary can propagate trust to more others than the friend who is more likely to be an adversary.

We denote the trust propagation capacity C_{ij} as the number of routers (note that one person could deploy multiple routers) to which v_i can propagate trust through the friend v_j. In our proposed adaptive trust propagation algorithm, C_{ij} must be proportionate to Φ_{ij}, such as:

$$C_{ij} = \left\lceil \frac{\Phi_{ij}}{\sum_{\Phi_{ik} \geqslant \Phi_h} \Phi_{ik}} \cdot \sum_{\Phi_{ik} \geqslant \Phi_h} C_{ik} \right\rceil. \tag{1}$$

where $\lceil * \rceil$ is a ceiling function. Φ_h is a group trust minimum threshold. SGor employs Φ_h to filter out ineligible friends because they are more likely to be wrongly trusted. $\sum_{\Phi_{ik} \geqslant \Phi_h} C_{ik}$ is the total number of routers to which v_i can propagate trust through eligible friends, while $\sum_{\Phi_{ik} \geqslant \Phi_h} \Phi_{ik}$ is the sum of group trust that v_i has in the eligible friends.

Using the adaptive trust propagation algorithm, if an adversary v_k can propagate v_i's trust, he can only propagate v_i's trust to at most C_{ik} other adversaries.

4.3. Trust graph based router selection

Based on group trust and global trust, SGor proposes a novel trust graph based router selection algorithm. This algorithm has three parameters to be set in advance. One is group trust minimum threshold Φ_h. The other two are R_i and L_i. The parameter R_i determines the number of candidate routers that a person v_i expects to collect through trust propagation. v_i constructs onion circuits by selecting routers uniformly at random from these R_i candidate routers. The parameter L_i limits the maximum number of hops to which v_i can propagate trust.

SGor implements the trust graph based router selection algorithm using a ticket distribution mechanism. This algorithm consists of three steps:

4.3.1. Initial step

When a person v_i (i.e., an onion routing user) attempts to construct onion circuits, he should create R_i tickets at first. Each ticket has a time-to-live field which specifies the maximum number of hops the ticket can be transmitted. v_i initializes this field to L_i. Afterwards, v_i distributes these R_i tickets to v_i's friends in whom v_i has group trust no smaller than Φ_h. The amount of tickets that can be distributed to each eligible friend is calculated according to Eq. (1), where $\sum_{\Phi_{ik} \geqslant \Phi_h} C_{ik} = R_i$.

4.3.2. Ticket distribution step

If a person v_j receives T_{ij} v_i's tickets, v_j consumes r_{ij} tickets to provide r_{ij} candidate routers to v_i. v_j determines r_{ij} depending on T_{ij} and r_j (i.e., r_j is the amount of onion routers that v_j deploys in SGor). If $T_{ij} \leqslant r_j$, $r_{ij} = T_{ij}$ and v_j stops the ticket distribution. Otherwise, $r_{ij} = r_j$. For the remaining $T_{ij} - r_j$ tickets, v_j first sets $L_i = L_i - 1$. If $L_i = 0$, v_j discards these remaining tickets and stops the ticket distribution. Otherwise, v_j will further distribute the remaining tickets to v_j's eligible friends who will repeat the ticket distribution step. The amount of tickets distributed to each of v_j's eligible friends is also computed using Eq. (1), where $\sum_{\Phi_{jk} \geqslant \Phi_h} C_{jk} = T_{ij} - r_j$. When distributing v_i's tickets, the ticket distribution step is performed by different people in parallel. Hence, some people could receive v_i's tickets more than one time (i.e., the duplicated ticket distribution).

To eliminate the duplicated ticket distribution, people should only distribute tickets to the eligible friends who have not received v_i's tickets before. The intuitive mechanism is to allow people querying their friends whether they have already received v_i's tickets before they distribute tickets. However, this mechanism is not resilient to attacks. Adversaries can simply claim they have never received v_i's tickets and repeatedly defraud tickets.

In this paper, we propose a *regressive checking mechanism* to effectively avoid both the duplicated ticket distribution and the ticket frauds. Using this mechanism, the ticket owner v_i maintains a list that records the people who have already received v_i's tickets. For each person v_j who attempts to distribute v_i's tickets, v_j should first check with v_i and exclude the friends who have already recorded in v_i's list. v_j then distributes v_i's tickets to v_j's remaining friends. Meanwhile, v_i adds these remaining friends to v_i's list.

4.3.3. Router discovery step

SGor has two candidate mechanisms for this step: (I) flooding-based router discovery; and (II) probabilistic router discovery.

Using the flooding-based router discovery, any person v_j who receives v_i's tickets should provide r_{ij} candidate routers to v_i. These candidate routers are sent to v_i alongside the backward path from v_j to v_i. Therefore, v_i can receive

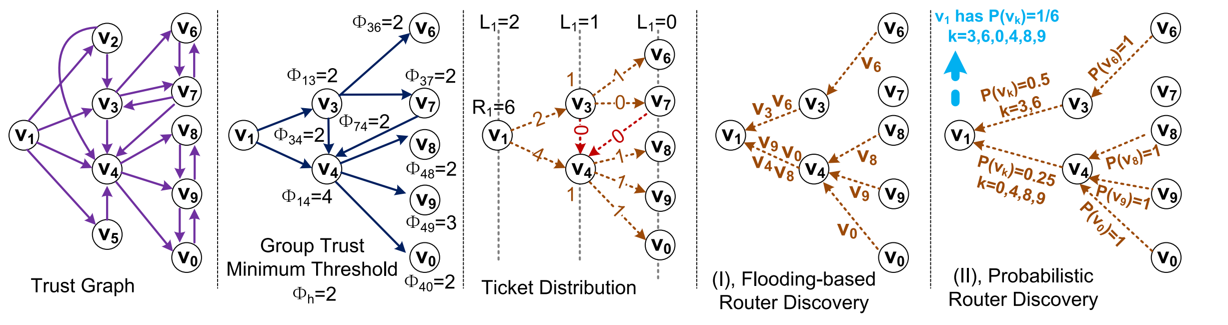

Trust Graph
Group Trust Minimum Threshold Φ_h=2
Ticket Distribution
(I), Flooding-based Router Discovery
(II), Probabilistic Router Discovery
v1 has P(vk)=1/6 k=3,6,0,4,8,9
P(vk)=0.5 k=3,6
P(vk)=0.25 k=0,4,8,9

Fig. 6. An example of trust graph based router selection.

$\sum_{v_j \in V_i} r_{ij} = R_i^*$ candidate routers in total, where V_i is the set of people who receive v_i's tickets. $R_i^* \leqslant R_i$ because some v_i's tickets could be discarded in the ticket distribution step. v_i constructs onion circuits by selecting routers uniformly at random from these R_i^* candidate routers, hence resulting in a selection probability $\frac{1}{R_i^*}$ for each candidate router. However, since all the R_i^* candidate routers are sent back alongside backward trust paths to v_i, the flooding-based router discovery runs a high risk of exposing honest routers to others.

To mitigate the exposure of honest routers, we propose a probabilistic router discovery mechanism. Using this mechanism, a person $v_j \in V_i$ can only response one candidate router to the person who distributes tickets to v_j. Let V_{ij} be the set of people who receive v_i's tickets from v_j (i.e., V_{ij} is the set of v_j's eligible friends who receive v_i's tickets). Let $P(v_k)$ be the probability that v_j sends back a router from v_k to v_j's predecessor, where $v_k \in V_{ij} \vee v_j$. The probabilistic router discovery sets $P(v_k)$ to $P(v_k) = \frac{r_{ik}}{\sum_{v_m \in V_{ij} \vee v_j} r_{im}}$. The same as the flooding-based router discovery, this probabilistic router discovery can also result in a selection probability $\frac{1}{R_i^*}$ for each candidate router. Since each person only sends back a single router to its predecessor, the probabilistic router discovery mitigates the chance of exposing honest routers to adversaries.

Fig. 6 demonstrates an example of the trust graph based router selection algorithm. In this example, each person is assumed to deploy a single router in SGor (i.e., $r_i = 1$, $i = 0, \ldots, 9$). v_1 is the person who attempts to construct onion circuits. The parameters are set as $\Phi_h = 2$, $R_1 = 6$ and $L_1 = 2$. In the initial step, v_1 generates $R_1 = 6$ tickets and set $L_1 = 2$ to the time-to-live field of these tickets. v_1 then distributes 2 tickets to v_3 and 4 tickets to v_4 according to Eq. (1) (i.e., $C_{13} = \lceil \frac{\Phi_{13}}{\Phi_{13}+\Phi_{14}} \rceil \cdot R_1 = 2$ and $C_{14} = \lceil \frac{\Phi_{14}}{\Phi_{13}+\Phi_{14}} \rceil \cdot R_1 = 4$). When v_3 and v_4 receive $T_{13} = C_{13} = 2$ and $T_{14} = C_{14} = 4$ tickets respectively, each of them consumes one because each person deploys a single router. For the remaining tickets (v_3 has $T_{13} - r_3 = 1$ and v_4 has $T_{14} - r_4 = 3$), since $L_1 = L_1 - 1 = 1 > 0$, v_3 and v_4 can further distribute these tickets to their eligible friends. v_3 has two eligible friends v_6 and v_7, while v_4 has three (v_8, v_9 and v_0). Although $\Phi_{34} = 2 \geqslant \Phi_h$, v_3 can exclude v_4 from the set of v_3's eligible friends using the regressive checking mechanism as v_4 has already received v_1's tickets from v_1.

Using Eq. (1), v_3 can calculate $C_{36} = \lceil \frac{\Phi_{36}}{\Phi_{36}+\Phi_{37}} \rceil \cdot (T_{13} - r_3) = 1$ and $C_{37} = \lceil \frac{\Phi_{37}}{\Phi_{36}+\Phi_{37}} \rceil \cdot (T_{13} - r_3) = 1$. However, v_3 only has $T_{13} - r_3 = 1$ remaining tickets. In this case, v_3 can simply decide to distribute 1 ticket to v_6 but 0 ticket to v_7, or vice versa. Upon receiving tickets, v_6, v_8, v_9 and v_0 stop the ticket distribution due to two reasons. One is $L_1 = L_1 - 1 = 0$, and the other is that they have no remaining tickets after they consume one. v_1 collects onion routers from the users who consume v_1's tickets. As can be seen in Fig. 6, both of the two router discovery mechanisms can result in $\frac{1}{6}$ selection probability for each candidate router, although v_3 and v_4 differently act in these two mechanisms. In the flooding-based router discovery, v_3 and v_4 simply send back all the routers they receive to v_1. While in the probabilistic router discovery, v_3 uses the probability 0.5 to choose v_3's router or v_6's router for responding v_1, and v_4 employs the probability 0.25 to chose responded routers.

We note that SGor is not the first one that uses ticket distribution mechanism. The Sybil-resilient rating system has already used it to defeat Sybil attacks [32]. However, SGor's ticket distribution is novel in several aspects. First, SGor distributes tickets according to group trust, hence having better capability of preventing tickets being distributed to adversaries. Second, since people consume tickets based on the number of routers they deploy, SGor can have better capability of evading adversaries' routers if honest people can deploy more routers. Third, to evade duplicated ticket distribution, the Sybil-resilient rating system requires a central server for calculating the shortest path in advance. Unlike that, SGor proposes a novel regressive checking mechanism to evade the duplicated ticket distribution. This checking mechanism does not require a centralized server.

4.4. SGor analysis

In this section, we analyze SGor's capability of protecting anonymity using a probabilistic model. We first discuss whether SGor can effectively evade adversaries' routers from users' onion circuits. We then investigate whether SGor can effectively defend against inference attacks.

4.4.1. The capability of evading adversaries' routers

Since SGor employs group trust and global trust to discover honest routers, we analyze how group trust and global trust affect the capability of evading adversaries' routers, respectively.

4.4.1.1. Group trust. In this analysis, we focus on answering the question how likely a person v_j is an adversary if an honest person v_i has group trust Φ_{ij} in v_j.

Let I_{ij} be the event that an honest user v_i assigns incorrect trust to an adversary v_j (i.e., $v_i \rightarrow v_j$ is an incorrect trust assignment). $P(I_{ij})$ is the probability that the event I_{ij} occurs. Since local trust assignments (i.e., trust edges) in SGor's trust graph $G = (V,E)$ are independent (see Sections 3.2 and 3.4), the events I_{ij} for different v_i, v_j are independent with each other. We therefore have $P(I_{ij}, I_{kl}) = P(I_{ij}) \cdot P(I_{kl})$ for $\forall v_i, v_j, v_k, v_l \in V$, where $P(I_{ij}, I_{kl})$ is the joint probability that both events I_{ij} and I_{kl} happen concurrently.

Let A_j be the event that the user $v_j \in V$ is an adversary. $P(A_j)$ is the probability that v_j is an adversary.

Let F_{ij} be the set of the people who are trusted by v_i and meanwhile trust v_j (i.e., for $\forall v_k \in F_{ij}$, there must exist $v_i \rightarrow v_k \rightarrow v_j$). Let $\Lambda_{ij} \subseteq F_{ij}$ be the set of people who are adversaries belonging to F_{ij}.

$P(\Phi_{ij} = N|A_j)$ is the probability that v_i has group trust $\Phi_{ij} = N$ for v_j on condition that v_j is an adversary. We can calculate $P(\Phi_{ij} = N|A_j)$ as:

$$P(\Phi_{ij} = N|A_j) = P(I_{ij}) \cdot \sum_{\Lambda_{ij} \subseteq F_{ij}} \prod_{v_k \in \Lambda_{ij}} P(A_k)P(I_{ik}) \cdot \prod_{v_k \in F_{ij} \setminus \Lambda_{ij}} (1 - P(A_k))P(I_{kj}). \quad (2)$$

where $P(I_{ij})$ is the probability that v_i assigns incorrect trust to the adversary v_j. $P(A_k)P(I_{ik})$ represents the probability that v_i assigns incorrect trust to v_k and v_k is another adversary who can forward v_i's trust to the adversary v_j. $(1 - P(A_k))P(I_{kj})$ is the probability that v_i assigns trust to an honest person v_k but v_k assigns incorrect trust to the adversary v_j.

Since $\Phi_{ij} = N$ indicates N robust trust paths from v_i to v_j (see Theorem 1), it is intuitive to have a corollary as below.

Corollary 1. *$P(\Phi_{ij} = N + 1|A_j) \leqslant P(\Phi_{ij} = N|A_j)$.*

Proof. Compared with $\Phi_{ij} = N$, v_i has one more robust trust path to v_j when $\Phi_{ij} = N + 1$. We can simply assume this additional path is $v_i \rightarrow v_k \rightarrow v_j$. Hence, we have:

$$P(\Phi_{ij} = N + 1|A_j) = (P(A_k)P(I_{ik}) + (1 - P(A_k))P(I_{kj})) \cdot P(\Phi_{ij} = N|A_j) \leqslant (P(A_k) + (1 - P(A_k))) \cdot P(\Phi_{ij} = N|A_j) = P(\Phi_{ij} = N|A_j). \quad \square$$

$P(A_j|\Phi_{ij} = N)$ is the probability that v_j is an adversary on condition that v_i has group trust $\Phi_{ij} = N$ for v_j. According to Bayes' theorem, we can have:

$$P(A_j|\Phi_{ij} = N) = \frac{P(\Phi_{ij} = N|A_j) \cdot P(A_j)}{P(\Phi_{ij} = N)}. \quad (3)$$

If the probabilities $P(A_j)$ and $P(\Phi_{ij})$ follow uniform distribution, $P(A_j|\Phi_{ij} = N)$ is proportionate to $P(\Phi_{ij} = N|A_j)$. Hence $P(A_j|\Phi_{ij} = N + 1) \leqslant P(A_j|\Phi_{ij} = N)$. That is to say, if an honest people v_i has a larger group trust in another people v_j, v_j is less likely to be an adversary. As a result, SGor has a better capability of evading adversaries' routers from users' onion circuits by using a larger Φ_h.

However, it is not the fact that the probabilities $P(A_j)$ and $P(\Phi_{ij})$ are uniformly distributed in practice. Hence, to show the effectiveness of group trust, we will do more practical evaluations using real-world datasets in Section 5.

4.4.1.2. Global trust. In this analysis, we focus on answering the question how likely a person v_j is an adversary if an honest person v_i can propagate global trust to v_j through a L_i-hop trust path.

If an honest person v_i propagates global trust to v_j through a L_i-hop trust path, the probability that v_j is an adversary can be calculated using a recursive function:

$$P(A_j|L_i) = P(A_k|L_i - 1) + (1 - P(A_k|L_i - 1)) \cdot P(A_j|\Phi_{kj}). \quad (4)$$

where v_k precedes v_j in the trust path from v_i to v_j. If v_k is an adversary, v_j must be an adversary. Otherwise, v_j has the probability $P(A_j|\Phi_{kj})$ to be an adversary. The base case of the recursive Eq. 4 is $P(A_m|L_i = 1) = P(A_m|\Phi_{im})$, where v_m is the first person after v_i in the trust path from v_i to v_j.

Since a larger L_i leads to a higher $P(A_j|L_i)$, a smaller L_i can help SGor achieve a better capability of evading adversaries' routers from users' onion circuits.

4.4.2. The capability of defeating inference attacks

To analyze SGor's capability of defeating inference attacks, we investigate how many users an onion router can be trusted and selected by in average. In SGor, global trust is used to guide people to trust more others in the trust graph, hence allowing honest routers to be trusted and selected by more users in return. We note that, if a router can be selected by more users, inference attackers are hard to guess the initial user of an onion circuit by observing this router.

SGor uses an adaptive trust propagation algorithm to derive global trust. This algorithm has two parameters L_i and Φ_h (see Section 4.3). L_i is used to limit the distance of trust propagation and Φ_h is the group trust minimum threshold that can be used to prevent trust propagation from adversaries.

Since a larger L_i and a smaller Φ_h could lead more routers' owners to be globally trusted, SGor obtains a better capability of thwarting inference attacks by using a larger L_i and a smaller Φ_h. However, as discussed in Section 4.4.1, a larger L_i and a smaller Φ_h also result in a worse capability of evading adversaries' routers. As a result, SGor should choose appropriate Φ_h and L_i to balance the capability of evading adversaries' routers and the capability of defeating inference attacks. Section 5 will evaluate SGor on top of real-world social trust datasets and show appropriate Φ_h and L_i for these datasets.

5. Evaluation

In this section, we evaluate SGor using two real-world social trust datasets. We first describe the datasets in Section 5.1. We then evaluate SGor's capability of evading adversaries' routers and the capability of defeating inference attacks in Sections 5.2 and 5.3, respectively. We compare SGor with other global trust-based routing protocols in terms of deanonymization expectation in Section 5.4. After evaluating the leakage of a priori trust relationships in Section 5.5, we also evaluate the overheads introduced by SGor in terms of storage and communication round trips in Section 5.6.

5.1. Datasets

We adopt two real-world social interaction graph datasets to evaluate SGor. One dataset contains post wall interactions between users in a New Orleans regional network in Facebook [19]. This interaction graph has 46,952 nodes and 876,993 trust edges. The other dataset is the collection of one month interactions from another Facebook anonymous regional network [20]. It includes 431,995 nodes and 781,862 trust edges.

In our evaluated interaction graphs, each node represents a person who registers an account in Facebook. Each trust edge indicates a local trust that one person assigns to another person (see Section 3.4). We assume any person can play the role as an onion routing user or an onion router owner, or both or none of them (see Section 3.5.1).

We preprocess these two datasets following a similar manner as X-Vine [33] and SybilLimit [34] did. In particular, we remove the nodes with low degrees (e.g., the sum of outdegree and indegree is less than 5). This process can guarantee all the nodes in our evaluation have reasonable connectivity to the trust graph. We also eliminate the self-linked edges (e.g., $v_i \rightarrow v_i$) because these trust edges could mislead group trust computation. After these appropriate preprocesses, we use the truncated datasets to evaluate SGor.

Table 2
The two interaction social graphs after preprocess.

Dataset	# of Nodes	# of Edges	Avg. degree	Graph density
New Orleans [19]	27,601	231,035	8.37	3×10^{-4}
Anonymous [20]	83,034	305,711	3.68	4.4×10^{-5}

Table 2 summarizes basic statistics of these two truncated datasets. The average degree takes both outdegree and indegree into consideration. It is calculated by using the number of edges to divide the number of nodes in the graph (i.e., $\frac{|E|}{|V|}$). The density of a graph is defined as the number of edges in the graph divided by the number of edges in a complete graph (i.e., $\frac{|E|}{|V|\cdot(|V|-1)}$). Both the average degree and graph density are graph properties that can be used to show whether a graph is well connected or not. A larger average degree and density leads to a more tight-knit graph.

To have a better understanding of how graph properties (i.e., average degree and graph density) affect the parameter selection in SGor (the selection of Φ_h and L_i), we generate six synthetic graphs using a tool NetworkX [35] for investigation. We put the results based on synthetic graphs in Appendix A.

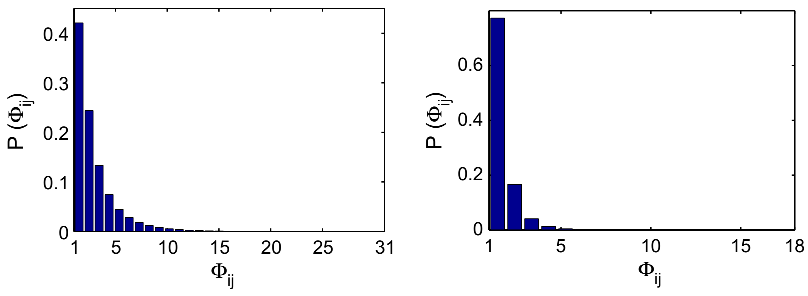
5.2. Evaluating the capability of evading malicious routers

In this section, we first evaluate group trust's effectiveness in evading adversaries' routers. We then show global trust's impacts to this effectiveness. Afterwards, we compare SGor with state-of-the-art trust-based onion routing to show SGor's improvement in protecting users from adversaries' routers.

5.2.1. Group trust's effectiveness

We measure group trust's effectiveness in evading adversaries' routers in terms of the probability $P(A_j|\Phi_{ij} = N)$. A smaller $P(A_j|\Phi_{ij} = N)$ indicates a smaller probability that v_j is an adversary given group trust $\Phi_{ij} = N$. According to Eq. (3), $P(A_j|\Phi_{ij} = N)$ is determined by $P(\Phi_{ij} = N)$, $P(\Phi_{ij} = N|A_j)$ and $P(A_j)$.

Based on the two real-world datasets listed in Table 2, we calculate practical distributions of $P(\Phi_{ij})$. In New Orleans dataset, Φ_{ij} ranges from 1 to 31. While in the anonymous dataset, $1 \leqslant \Phi_{ij} \leqslant 18$. Fig. 7 illustrates histograms of $P(\Phi_{ij})$ for the two datasets. It can be seen that the two datasets have the similar distributions of Φ_{ij}. A larger Φ_{ij} results in a smaller $P(\Phi_{ij})$. Moreover, there are more Φ_{ij}s having a smaller value in the Anonymous dataset than those in the New Orleans dataset. This is probably due to the lower graph density in the Anonymous dataset. We have further investigated this phenomenon using syntactic graphs in Appendix A.



(a) New Orleans Dataset [19]. (b) Anonymous Dataset [20].

Fig. 7. The real-world group trust distributions of $P(\Phi_{ij})$.

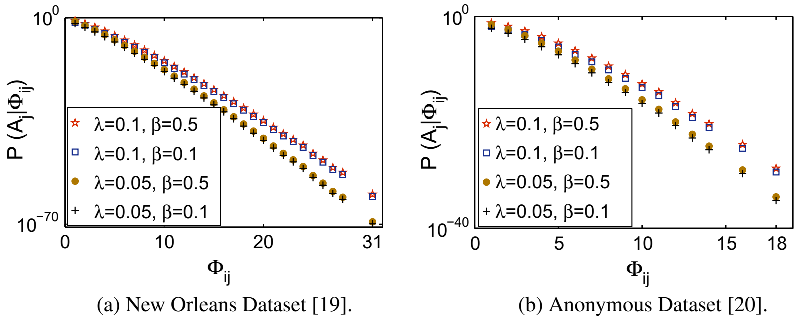


Fig. 8. The probability $P(A_j|\Phi_{ij})$ with different Φ_{ij} in four settings of λ and β.

According to Theorem 1 and Eq. (2), $P(\Phi_{ij} = N|A_j)$ is determined by the probabilities of events that honest people assign incorrect trust to adversaries. Since incorrect trust assignments are human behaviors, we can model these behaviors as a Poisson process with an average incorrect trust assignments probability λ (i.e., $P(\Phi_{ij} = N|A_j)$ follows a Poisson distribution with a parameter λ). A large λ indicates that a large number of people in SGor assign incorrect trust to adversaries. Since it is very difficult for adversaries to attract incorrect trust from a large number of honest human beings [26], SGor is not designed for a large λ. Therefore, we only consider a small λ in our evaluation.

We consider adversaries as independent and identically distributed, i.e., the events A_j are i.i.d. In this case, $P(A_j)$ for $\forall v_j \in V$ is equal to a constant $\beta \in (0,1)$. A large β indicates a large fraction of the network is compromised by adversaries.

We evaluate the effectiveness of group trust Φ_{ij} in protecting an honest person v_i from an adversary v_j's routers by investigating $P(A_j|\Phi_{ij})$ for different Φ_{ij}s in the two real-world datasets. A smaller $P(A_j|\Phi_{ij})$ means a local trust assignment $v_i \to v_j$ is less likely to be incorrect, hence indicating a better capability of evading adversaries' routers. In this evaluation, we consider $\lambda = 0.1$ or $\lambda = 0.05$ and $\beta = 0.5$ or $\beta = 0.1$. Fig. 8 shows the evaluation results. It can be seen that a larger Φ_{ij} leads to a smaller $P(A_j|\Phi_{ij})$ (i.e., a better capability of evading adversaries). In particular, for the dataset [19], $P(A_j|\Phi_{ij})$ drops about 10^{68} times when the group trust climbs from $\Phi_{ij} = 1$ to 31. For the dataset [20], $P(A_j|\Phi_{ij})$ declines around 10^{35} times when the group trust grows up from $\Phi_{ij} = 1$ to 18. Moreover, $P(A_j|\Phi_{ij})$ with a setting of smaller λ and β is smaller than that with a setting of larger λ and β. This indicates that group trust is more effective in evading adversaries when honest people make less incorrect local trust assignments in SGor and adversaries control a smaller fraction of the SGor network.

5.2.2. Global trust's impact

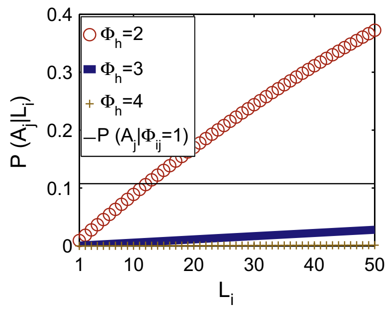
We evaluate the global trust's impacts on the capability of evading adversaries' routers by investigating $P(A_j|L_i)$. A smaller $P(A_j|L_i)$ indicates that v_j is less likely to be an adversary if an honest person v_i propagate global trust to v_j through a trust path consisting of L_i trust edges. When SGor derives global trust using adaptive trust propagation algorithm, $P(A_j|L_i)$ is determined by the length of the trust path L_i and the group trust of each trust edge in the trust path. The group trust of each trust edge must be no smaller than a minimum threshold Φ_h (see Section 4.3). A larger L_i results in a larger $P(A_j|L_i)$, while a larger Φ_h leads to a smaller $P(A_j|L_i)$.

To evaluate global trust's impact, we investigate $P(A_j|L_i)$ when L_i varies in different cases of Φ_h. We compare $P(A_j|L_i)$ with $P(A_j|\Phi_{ij} = 1)$. $P(A_j|\Phi_{ij} = 1)$ represents how likely v_j is an adversary if v_i has local trust in v_j. $P(A_j|L_i) < P(A_j|\Phi_{ij} = 1)$ indicates that the people who receive global trust through a trust path consisting of L_i trust edges are less likely to be adversaries than the people who receive local trust. As a result, if SGor can achieve $P(A_j|L_i) < P(A_j|\Phi_{ij} = 1)$ using appropriate L_i and Φ_h, the global trust could not degrade the capability of evading adversaries' routers.

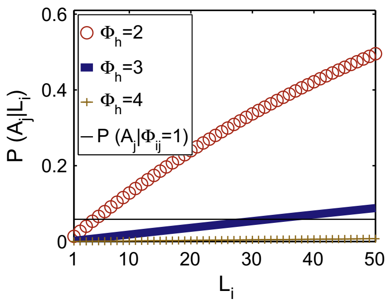
Fig. 9 shows the probability $P(A_j|L_i)$ when $\Phi_h = 2$, 3, 4 and L_i increases from 1 to 50. In this evaluation, we use the setting of $\lambda = 0.1$ and $\beta = 0.5$, and consider the worst case that the trust propagation is through the trust paths where any trust edges $v_k \to v_m$ have $\Phi_{km} = \Phi_h$. In the dataset [19], for the case $\Phi_h = 2$, $P(A_j|L_i)$ becomes larger than $P(A_j|\Phi_{ij} = 1)$ when $L_i > 12$. For the other cases $\Phi_h = 3$ and $\Phi_h = 4$, $P(A_j|L_i)$ remains smaller than $P(A_j|\Phi_{ij} = 1)$ even if L_i grows up to 50. While in the dataset [20], for the cases $\Phi_h = 2$ and $\Phi_h = 3$, $P(A_j|L_i)$ is larger than $P(A_j|\Phi_{ij} = 1)$ when $L_i > 5$ and $L_i > 30$ respectively. Based on this evaluation, to maintain the capability of evading adversaries' routers, SGor should propagate global trust through the trust paths no longer than $L_i = 12$ if $\Phi_h = 2$ in the dataset [19], and no longer than $L_i = 5$ if $\Phi_h = 2$ or $L_i = 30$ if $\Phi_h = 3$ in the dataset [20]. It can be seen, SGor can choose a larger L_i for trust propagation in the graph with higher density (i.e., the dataset [19] has a higher graph density than the dataset [20]).

5.2.3. Simulation of SGor and trust-based onion routing

To show SGor's advantage in evading adversaries' routers compared with trust-based onion routing, we simulate router selections based on the two real-world datasets with setting of $\lambda = 0.1$ and $\beta = 0.5$. In this simulation, SGor sets $L_i = 1$, $\Phi_h = 2$, 3, 4 and R_i to a large enough value, respectively. SGor uses Φ_h to filter the routers whose group trust is smaller than Φ_h and propagates global trust to at most L_i hops. We note that the trust-based onion routing operates in the same manner as SGor with $L_i = 1$ and $\Phi_h = 1$. We also assume each person deploys a single router in the network. We generate a pseudo random value

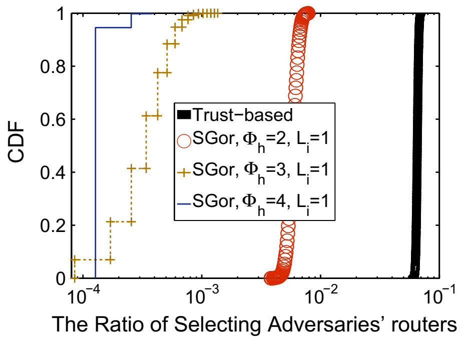


(a) New Orleans Dataset [19].

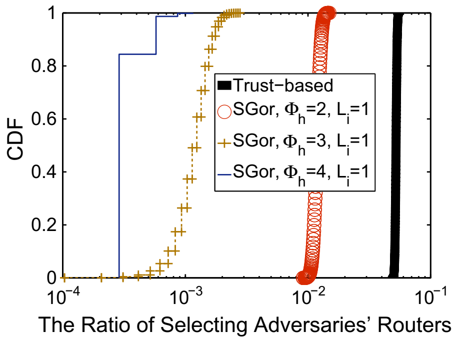


(b) Anonymous Dataset [20].

Fig. 9. The probability $P(A_j|L_i)$ when L_i is from 1 to 50.



(a) New Orleans Dataset [19].



(b) Anonymous Dataset [20].

Fig. 10. The CDFs of the ratio of selecting adversaries' routers in 10,000 rounds of router selection simulation in SGor and trust-based onion routing.

$\pi \in [0,1]$ for each person v_j who deploys an onion router. From an honest person v_i's point of view, v_j is regarded as an adversary if $\pi < P(A_j|L_i)$.

We simulate a person v_i as an honest person to select a router deployed by other people using existing trust-based onion routing algorithms [2,3] or the trust graph based algorithm proposed in Section 4.3. We define a round of simulation by iterating all the people in the two real-world datasets to operate as v_i one by one. We conduct 10,000 rounds of router selection simulations for trust-based onion routing and SGor with $L_i = 1$, $\Phi_h = 2$, 3, 4, respectively.

In each round of simulation, we measure the capability of evading adversaries' routers using the ratio of selecting adversaries' routers. This ratio is calculated by using the number of people who select adversaries' routers in each round to divide the total number of people. A smaller ratio means a better capability of evading adversaries' routers.

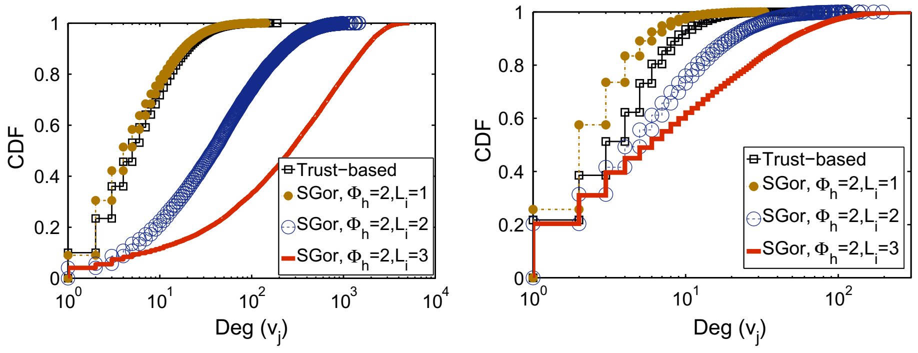
Fig. 10(a) and (b) show the cumulative distribution function (CDF) of the ratio of selecting adversaries' routers in our simulations. It can be seen that SGor outperforms trust-based onion routing in both the New Orleans dataset [19] and the anonymous dataset [20]. In particular, SGor with parameters $\Phi_h = 2$ and $L_i = 1$ has more than 10 times improvement for evading adversaries' routers. When the parameter Φ_h is increased to 4, the improvement extends to around 1000 times.

5.3. Evaluating the capability of defeating inference attacks

We evaluate an honest onion router's capability of defeating inference attacks by investigating the number of users who can select this router. A network (SGor or trust-based onion routing) has a better capability of resisting inference attacks if more honest onion routers can be selected by more users.

Let $Deg(v_j)$ be the number of people who can select the onion routers deployed by v_j. In trust-based onion routing, a person v_i can select another person v_j's routers if and only if v_i has local trust in v_j (i.e., $v_i \rightarrow v_j$ exists in G). Hence, $Deg(v_j)$ is the in-degree of the node v_j in the trust graph G. While in SGor, v_i can select v_j's routers in two cases. One is $\Phi_{ij} \geqslant \Phi_h$ if $v_i \rightarrow v_j$ exists. The other is v_i can propagate global trust to v_j using the adaptive trust propagation algorithm described in Sections 4.2 and 4.3.

We use $Deg(v_j)$ as a measure to compare the capability of defeating inference attacks between SGor and trust-based onion routing. A larger $Deg(v_j)$ indicates a better capability of defeating inference attacks. We consider SGor with parameters $L_i = 2$, 3 and $\Phi_h = 2$. We also choose R_i as a value which is large enough to guarantee SGor can propagate trust to $L_i = 2$ or 3. Our evaluation adopts the two real-world datasets listed in Table 2 and considers all the people to be as v_j one by one.



(a) New Orleans Dataset [19]. (b) Anonymous Dataset [20].

Fig. 11. The CDFs of $Deg(v_j)$ for SGor and trust-based onion routing.

Fig. 11 shows the CDF of $Deg(v_j)$ for SGor and trust-based onion routing in our evaluation. Compared to trust-based onion routing, SGor with $\Phi_h = 2$ and $L_i = 1$ suffers a distribution with more slightly small $Deg(v_j)$ s because $\Phi_h = 2$ filters trust edges whose group trust is smaller than 2. This slight degradation does not make a significant impact on the capability of defeating inference attacks, because these removed trust edges are likely to be incorrect trust assignments. When L_i is increased to 2, SGor reaches a distribution with more large $Deg(v_j)$ s than trust-based onion routing. The value of $Deg(v_j)$ is further increased when L_i grows up to 3 in SGor. These results show that SGor with a larger L_i can have a better capability of defeating inference attacks. By referring back to Fig. 9, we note that SGor with $\Phi_h \geqslant 2$ and $L_i \leqslant 5$ retains a better capability of evading adversaries' routers than trust-based onion routing. As a result, SGor apparently demonstrates a better capability of defeating inference attacks and evading adversaries' routers simultaneously.

5.4. Comparing SGor with other global trust-based schemes

Although Sections 5.2 and 5.3 have proved SGor's better performance of protecting anonymity than trust-based routing approaches that utilize local trust [1–3,15], we need to further compare SGor with prior studies which also apply global trust to onion routing. In the literature, there are two projects, Drac [17] and Pisces [18], that fall into this category. However, since Pisces has been proved to have better capability of protecting anonymity than Drac [18], we only compare SGor with Pisces because Pisces represents the state-of-the-art techniques using global trust.

To have a fair comparison between SGor and Pisces, we propose the use of deanonymization expectation to measure the anonymity each user can obtain from the two schemes. This measure takes both the capabilities of evading adversaries' routers and defeating inference attacks into account.

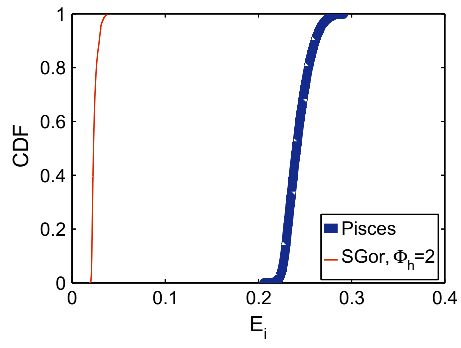
Let E_i be the person v_i's deanonymization expectation. E_i can be calculated as $E_i = \sum_{v_j \in \mathcal{F}_i} P_{ij} \cdot E_{ij}$. Where $\mathcal{F}_i$ is the set of persons whose router can be selected by v_i. P_{ij} is the probability that the person v_i uses to select the person v_j's router. It is determined by different trust-based routing algorithms. E_{ij} is the average probability that v_i can be deanonymized due to the selection of v_j's router. We have $E_{ij} = P(A_j|L_i) \cdot 1 + (1 - P(A_j|L_i)) \cdot P(v_i|v_j) \cdot \mu$. The constant 1 in $P(A_j|L_i) \cdot 1$ is the probability that v_i can be deanonymized when v_j is an adversary. This probability equals to 1 because the adversary v_j can launch correlation-like attacks to deanonymize v_i directly. $P(v_i|v_j)$ represents the probability that adversaries can guess v_i by observing v_j's router (i.e., inference attacks) when v_j is not an adversary. μ is the probability that adversaries can observe routers in onion circuits. We choose a small $\mu = 0.01$ in our evaluation, because adversaries can only observe routers by exploiting adversaries' routers or web servers (see Section 3.3), but the web servers adversaries can control are relatively few and users rarely select adversaries' routers for onion routing after using trust-based algorithm (see Fig. 8). L_i which is the number of hops v_i can propagate trust in SGor equals to the number of random walk steps in Pisces. Apparently, a smaller E_i indicates a better capability of protecting anonymity.

In our experiments, we compare SGor and Pisces by considering the setting of $\lambda = 0.1$ and $\beta = 0.5$. We choose $\Phi_h = 2$ for SGor. Moreover, we consider $L_i = 2$ and $L_i = 3$ in our comparison. That is, SGor propagates trust through 2 or 3-hop trust path while Pisces launches 2 or 3-step metropolis–hastings random walks on top of the New Orleans dataset [19] and the Anonymous dataset [20], respectively. Although we cannot guarantee the random walk can globally converge within $L_i = 2$ or $L_i = 3$ steps (it depends on whether the graph is fast-mixing), Pisces employs metropolis–hastings algorithm to guarantee the reach probability of each router during the random walk is approximately uniform distributed [18].

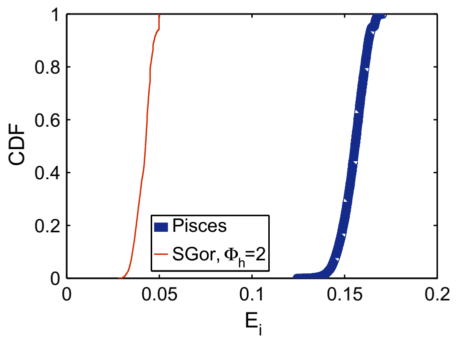
Fig. 12 shows the results of our comparison. Apparently, SGor achieves a better capability in protecting anonymity (in terms of a smaller deanonymizaiton expectation E_i). SGor outperforms Pisces because SGor employs group trust to rate limit trust propagation. This idea can effectively prevent adversaries from compromising global trust in trust propagation.

5.5. Evaluating the leakage of a priori trust relationships

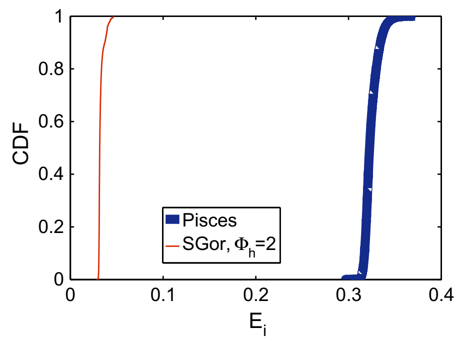
Since SGor is designed to run in a fully decentralized manner, it is not required to leak a priori trust



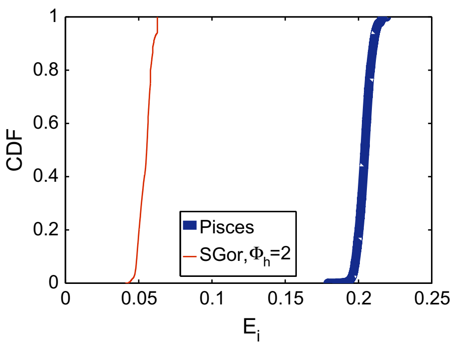
(a) Dataset [19] when $L_i = 2$.



(b) Dataset [20] when $L_i = 2$.

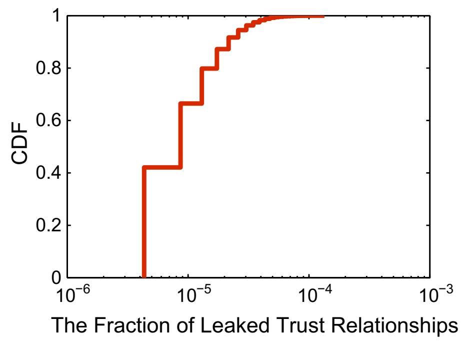


(c) Dataset [19] when $L_i = 3$.

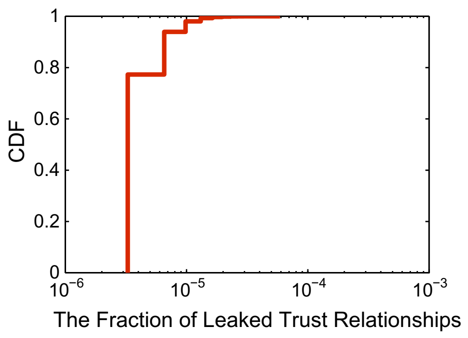


(d) Dataset [20] when $L_i = 3$.

Fig. 12. The CDF of E_i in Pisces and SGor with $\Phi_h = 2$.



(a) New Orleans Dataset [19].



(b) Anonymous Dataset [20].

Fig. 13. The CDF of the fraction of trust relationships that are leaked to each third person who calculates group trust.

relationships to any third parties, such as a centralized server. However, as the calculation of group trust is based on mutual friends information, people's friends could expose some of their trust edges to a third person who computes group trust. For example, if v_i has two robust trust paths to v_j, one is $v_i \rightarrow v_j$ and the other is $v_i \rightarrow v_k \rightarrow v_j$, to calculate Φ_{ij}, v_k's trust edge $v_k \rightarrow v_j$ could be leaked to v_i. As a result, SGor could leak some of people's trust relationships to other people.

We measure the trust relationships leaked to a third person in terms of the fraction of leaked trust edges. The fraction can be calculated by using the number of trust edges that are leaked to an individual third person during group trust calculation to divide the total number of trust edges in SGor's trust graph. A larger fraction indicates a larger amount of trust edges leaked.

Fig. 13 shows the CDF of the fraction of trust relationships that are leaked to each third person who computes group trust. It can be seen that the leaked fraction is negligible (less than 0.0002). Although a large number of adversaries could collude with each others (i.e., Sybil attacks), they cannot effectively enlarge the fraction of leaked trust relationships, because adversaries should compromise trust with two honest people to leak these two people's trust relationship through group trust computation. However, it is very difficult for adversaries to compromise trust with a large number of honest people. Previous Sybil defences (e.g., SybilLimit [34]) are usually designed based on this observation. As a result, the adversaries who perform inference attacks cannot benefit a lot from the decentralized design of SGor when they collect a priori trust relationships.

5.6. Evaluating SGor's overheads

To support trust graph based onion routing, SGor introduces overheads because it performs additional storage and communication in the trust graph layer. In this section, we evaluate these overheads in terms of storage in Section 5.6.1, communication round trips in Section 5.6.2, and additional traffic in Section 5.6.3. The results confirm that SGor only induces very few overheads due to its decentralized design.

5.6.1. Storage overheads

Benefit from the decentralized design, SGor is not required to deploy a centralized server to manage the entire trust graph. Instead, each person in SGor has a trust table to store his own local trust assignments, i.e., trust edges from this person to this person's friends (see Section 3.5.2). As a result, the storage overheads introduced by SGor are due to the additional storage space required by the trust table.

We evaluate the storage overheads by investigating the number of trust edges stored in each person's trust table. Fig. 14(a) and (b) plot the CDF of this number for the people in datasets [19,20], respectively. It can be seen that most of people (around 80%) have less than 20 trust edges stored in their trust table. Compared to the total number of trust edges in these two datasets (as shown in Table 2, the total number of trust edges is 231,035 in dataset [19] and 305,711 in dataset [20]), the number of trust edges stored in each person's trust table is relatively small. As a result, SGor introduces a few storage overheads for each person.

5.6.2. Communication overheads

As described in Section 4.3, onion routing users need to run a trust graph based algorithm to discover honest routers in SGor. This algorithm induces additional communications among people (through people's user-agents) in the trust graph layer.

We evaluate SGor's communication overheads in terms of communication round trips. A communication round trip is the time delay of a bi-directional communication between two people in the trust graph layer. Since SGor operates in a fully decentralized manner, most communications between different two people can be performed in parallel. According to SGor's decentralized algorithms (see Sections 4.1.3 and 4.3), the number of additional communication round trips introduced by SGor can be calculated by summing up one and a half round trips (for group trust computation) and the length of the longest trust path through which SGor propagates global trust (i.e., $L_i + 1.5$). These additional communication round trips are only determined by the parameter L_i and cannot be affected by the size of trust graph. Since $L_i + 1.5$ is a very small value compared with the size of a trust graph, SGor introduces a few communication overheads even if it adopts a large scale trust graph.

We investigate the number of parallel communications when people run trust graph based router selection algorithm in SGor. This number indicates the benefits that SGor can obtain from the decentralized design to reduce communication overheads. A larger number of parallel communications means more efficiency benefits.

Fig. 15 shows the CDFs of the number of parallel communications during different number of communication round trips. In this evaluation, people are simulated to run decentralized ticket distribution algorithm to discover honest routers on top of the datasets [19,20]. We consider $\Phi_h = 2$ and $L_i = 3$. It can be seen that more communication round trips could result in more parallel communications. This is the reason why a large scale trust graph does not introduce a serious communication overhead in SGor.

5.6.3. Additional traffic

In contrast to the additional communications, SGor also introduces additional traffic to the network. This overhead is caused by group trust aggregation and adaptive global trust propagation.

When SGor aggregates group trust using the decentralized token-based algorithm (see Section 4.1.3), we measure the additional traffic in terms of the number of tokens transmitted in each trust edge. When a person

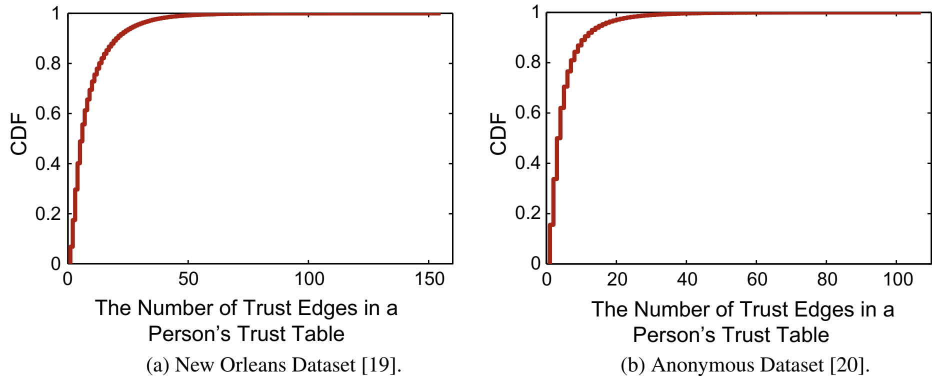
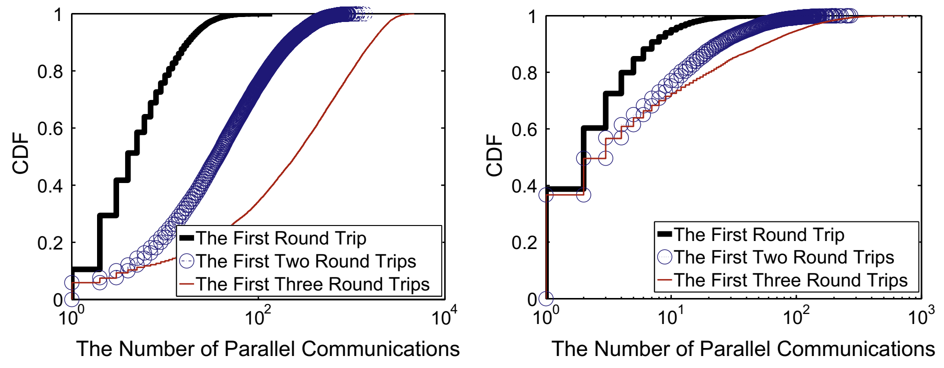
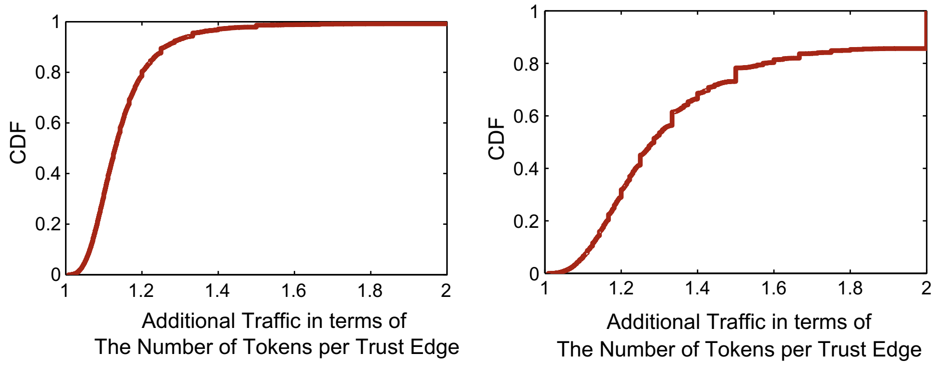


Fig. 14. The CDF of the number of trust edges stored in each person's trust table in SGor.



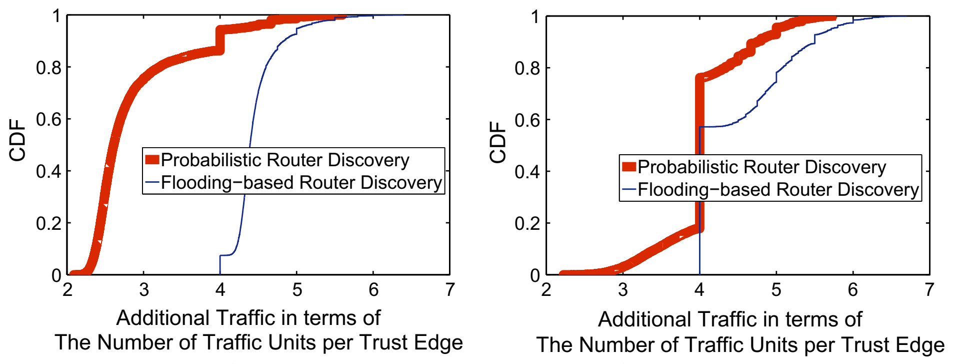
(a) New Orleans Dataset [19]. (b) Anonymous Dataset [20].

Fig. 15. The CDFs of the number of parallel communications during different number of communication round trips in SGor with $\Phi_h = 2$ and $L_i = 3$.



(a) New Orleans Dataset [19]. (b) Anonymous Dataset [20].

Fig. 16. The CDFs of additional traffic introduced by group trust aggregation (in terms of the number of tokens per trust edge) in SGor.



(a) New Orleans Dataset [19]. (b) Anonymous Dataset [20].

Fig. 17. The CDFs of additional traffic introduced by global trust propagation (in terms of the number of traffic units transmitted in each trust edge) in SGor when $\Phi_h = 2$ and $L_i = 3$.

calculates group trust for his friends, two kinds of trust edges are involved. One includes the edges from this person to his friends, and the other includes the edges from this person's friends to the friends of this person's friends. The tokens can be distributed through the first kind of edges at first, and then further forwarded via the second kind of edges. At last, appropriate tokens can be sent back to the initial user through the first kind of edges again. The number of tokens transmitted in each trust edge is computed by using the number of tokens transmitted on the fly to divide the number of trust edges involved.

Fig. 16 illustrates the CDF of the average number of tokens transmitted in each trust edge. Apparently, the additional traffic introduced by group trust calculation is not

Table A.3
Statistics in six syntactic graphs.

Dataset	# of Nodes	# of Edges	Avg. degree	Graph density
Syntactic Graph ①	1000	4975	4.98	5×10^{-3}
Syntactic Graph ②	1000	10,879	10.9	1.1×10^{-2}
Syntactic Graph ③	1000	51,975	52.0	5.2×10^{-2}
Syntactic Graph ④	1000	97,900	98.0	9.8×10^{-2}
Syntactic Graph ⑤	2000	9975	5.00	2.5×10^{-3}
Syntactic Graph ⑥	2000	106,975	53.5	2.7×10^{-2}

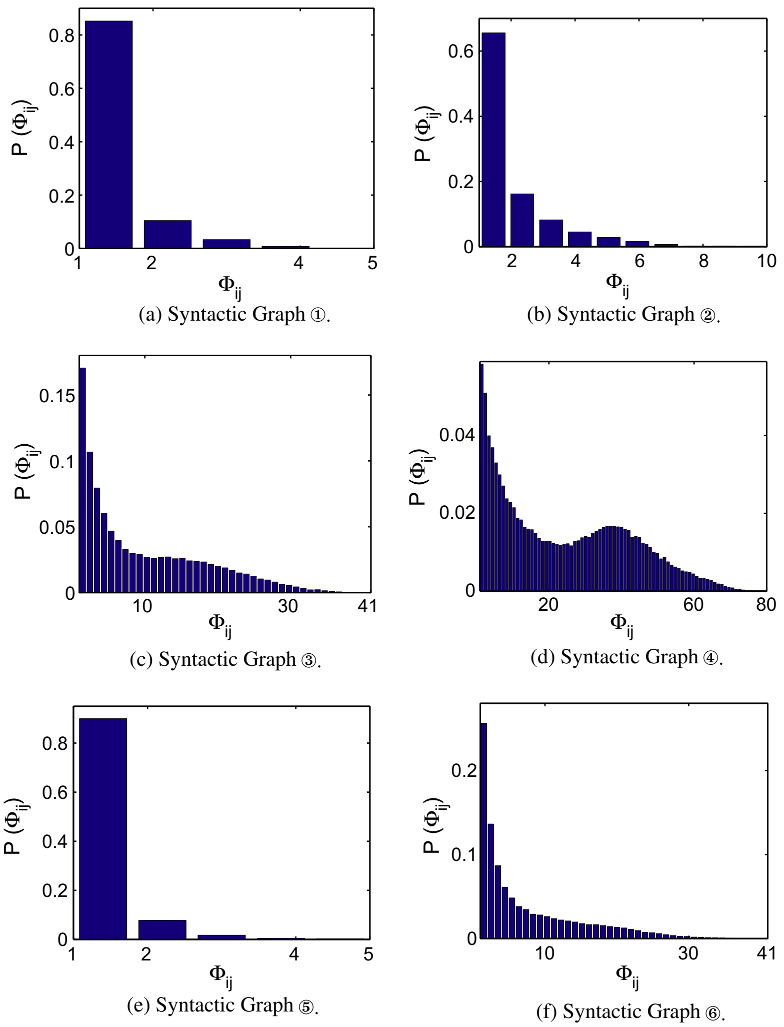


Fig. A.18. The group trust distributions of $P(\Phi_{ij})$ in syntactic graphs.

large because no more than 2 tokens per trust edge are inserted into the network. However, there is an interesting observation that the New Orleans dataset introduces less additional traffic than the Anonymous dataset, although the group trust in the former dataset is statistically larger than that in the latter dataset. The reason of this observation is because a large number of people in the Anonymous dataset have none friends even if they are friends of other people. These people cannot induce the second kind of trust edges during group trust calculation, hence resulting a small number of trust edges when we compute the number of tokens per trust edge.

Besides group trust aggregation, the adaptive global trust propagation also injects additional traffic into the network. We measure this kind of additional traffic in terms of traffic units that are conveyed in each trust edge.

We use traffic unit as the measure, because global trust propagation could induce three different kinds of additional traffic. As described in Section 4.3, our global trust propagation algorithm consists of a tickets forwarding, a regressive checking mechanism for evading duplicated ticket distribution and a router collection. Each tickets forwarding injects one unit of additional traffic as it only forwards the number of remaining tickets through trust edges. For the regressive checking mechanism, the checking request and the checking response consume one traffic unit each. For the router collection, each router transmitted in the trust edge induces one additional traffic unit.

Fig. 17 shows the CDFs of the number of traffic units transmitted in each trust edge during global trust propagation in SGor when $\Phi_h = 2$ and $L_i = 3$. It can be seen that the additional traffic is relatively small (no more than 7 traffic units are transmitted in each trust edge). Moreover, although the probabilistic router discovery mechanism is originally proposed to mitigate the chances of router exposure during trust propagation, our result confirms that this mechanism is also effective in reducing additional traffic compared with the flooding-based router discovery.

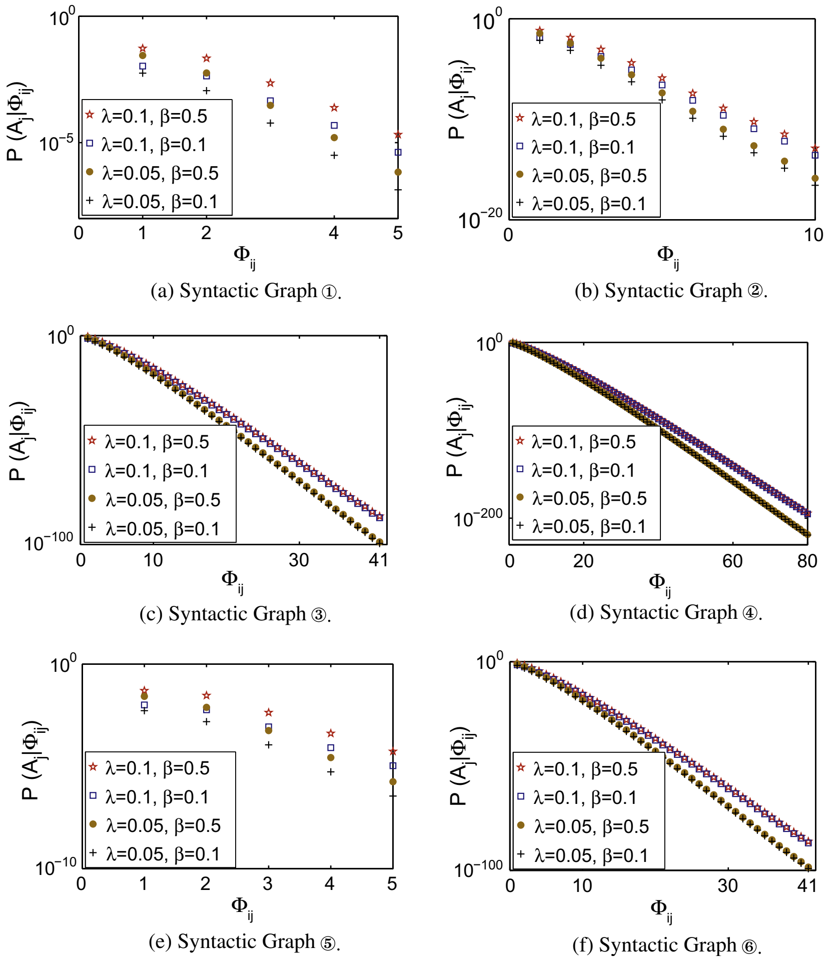
6. Discussion and future work

Although SGor successfully mitigates key limitations in trust-based onion routing, there is still room for further improvement:

First, if the trust graph that SGor adopts cannot provide enough robust trust paths to form group trust, SGor would degenerate into the traditional trust-based onion routing. For example, only 69.7% people in the interaction graph from the dataset [19] have friends who can receive group trust larger than $\Phi_h = 2$. In this case, the other 30.3% people can only use SGor as the traditional trust-based onion routing. Enlarging the population of SGor is an interesting future work.

Second, since we only evaluate SGor with independent and identically distributed adversaries in this paper, we leave a future work to assess SGor using other adversary distributions. For example, to show SGor's effectiveness in the presence of practical adversaries, we need to sample adversary distributions from real-world datasets.

Third, we design SGor by considering a stable trust graph. However, users are usually dynamic because they



(a) Syntactic Graph ①. (b) Syntactic Graph ②.

(c) Syntactic Graph ③. (d) Syntactic Graph ④.

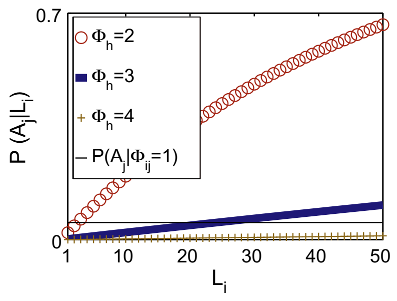
(e) Syntactic Graph ⑤. (f) Syntactic Graph ⑥.

Fig. A.19. The probability $P(A_j|\Phi_{ij})$ with different Φ_{ij} in four settings of λ and β.

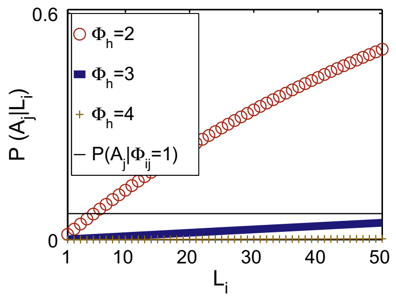
could join and leave SGor frequently. We argue that users' trust relationships are more stable. Instead, their dynamics are usually in the form of frequent online and offline. To address this issue, we suggest users to run their user-agents in an uninterrupted service, such as a virtual machine in EC2 or PlanetLab. If some users fail to run their agents in uninterrupted service, the resulted trust graph is incomplete. The incomplete graph could cause live users missing to discover some live honest routers. To address this problem, a possible solution is to fetch the missing part of trust graph in a centralized service. But this solution affects SGor's decentralized scalability. As a result, there is a tradeoff between scalability and graph completeness, and this tradeoff can be made differently, depending on the system's objectives.

Fourth, as described in Section 3.2, SGor is designed based on an assumption that all the trust edges in the trust graph are independent. Although we argue that trust edges in an interaction graph are more independent, we also encourage SGor to build up a customized trust network. In this trust network, onion routing users and routers' owners are enabled to evaluate the trustworthiness of each other independently.

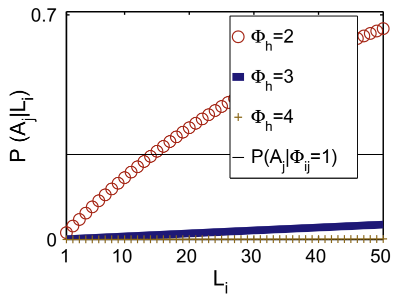
Fifth, we acknowledge that SGor, like all existing trust-based routing systems (e.g., Drac and Pisces), is still subject to social engineering attacks to some extent. For example, if adversaries can launch social engineering attacks to defraud some honest people who have low security awareness, they could arbitrarily act to make impact to these victims in our proposed scheme. Fortunately, due to three reasons, this impact is relatively small: ① The trust levels that adversaries receive through social engineering attacks are usually low. ② Mutual friends based group trust can mitigate this attack. ③ Our proposed adaptive trust propagation algorithm has the capability of limiting the influence of this attack. However, we also agree this is a weakness of current SGor and expect countermeasures in the future.



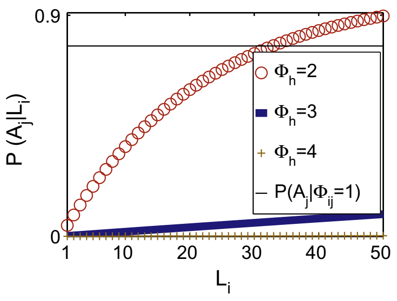
(a) Syntactic Graph ①.



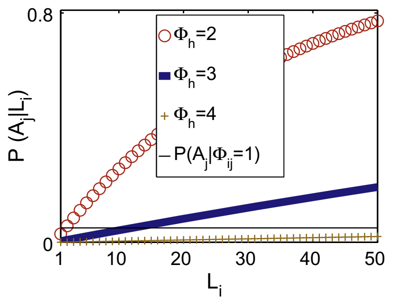
(b) Syntactic Graph ②.



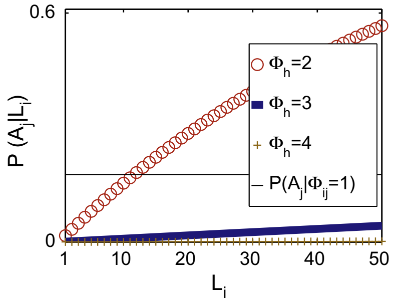
(c) Syntactic Graph ③.



(d) Syntactic Graph ④.



(e) Syntactic Graph ⑤.



(f) Syntactic Graph ⑥.

Fig. A.20. The probability $P(A_j|L_i)$ when L_i is from 1 to 50.

Table A.4
The selection of L_i given a Φ_h for SGor to guarantee $P(A_j|L_i) < P(A_j|\Phi_{ij} = 1)$ in different Syntactic Graphs.

Dataset	Avg. degree	Graph density	Φ_h	L_i
Syntactic Graph ①	4.98	5×10^{-3}	$\Phi_h = 2$	$L_i \leqslant 2$
			$\Phi_h = 3$	$L_i \leqslant 26$
			$\Phi_h = 4$	L_i up to 50
Syntactic Graph ②	10.9	1.1×10^{-2}	$\Phi_h = 2$	$L_i \leqslant 5$
			$\Phi_h = 3$	L_i up to 50
			$\Phi_h = 4$	L_i up to 50
Syntactic Graph ③	52.0	5.2×10^{-2}	$\Phi_h = 2$	$L_i \leqslant 14$
			$\Phi_h = 3$	L_i up to 50
			$\Phi_h = 4$	L_i up to 50
Syntactic Graph ④	98.0	9.8×10^{-2}	$\Phi_h = 2$	$L_i \leqslant 33$
			$\Phi_h = 3$	L_i up to 50
			$\Phi_h = 4$	L_i up to 50
Syntactic Graph ⑤	5.00	2.5×10^{-3}	$\Phi_h = 2$	$L_i \leqslant 1$
			$\Phi_h = 3$	$L_i \leqslant 14$
			$\Phi_h = 4$	L_i up to 50
Syntactic Graph ⑥	53.5	2.7×10^{-2}	$\Phi_h = 2$	$L_i \leqslant 11$
			$\Phi_h = 3$	L_i up to 50
			$\Phi_h = 4$	L_i up to 50

7. Conclusions

In this paper, we explore new trust features from trust graph and show promising solutions to address the fundamental challenges in state-of-the-art trust-based onion routing. Although we consider only the group trust and global trust in our prototype design, we expect future research can benefit from our research principle and discover more useful trust features to further improve the protection of anonymity in onion routing networks.

Appendix A. Syntactic graph based analysis

Our syntactic graph based analysis is presented here to investigate how graph properties (e.g., average degree and density) affect the parameter (e.g., Φ_h and L_i) selection in SGor. In particular, we generate syntactic graphs by calling the function `barabasi_albert_graph ()` in NetworkX [35]. This function returns a random scale-free graph based on a Barabási-Albert preferential attachment model [36]. The trust graphs following this model are widely observed in nature and human-made systems, which include social networks.

We have generated six syntactic graphs with different sizes, average degrees and graph densities. Table A.3 summarizes the statistics of these six graphs.

We show the group trust (i.e., $P(\Phi_{ij})$) distributions of these six graphs in Fig. A.18(a)–(f). By analyzing these figures as well as consulting Table A.3, we confirm that a larger average degree or a higher graph density results in a larger value of $P(\Phi_{ij})$. Moreover, compared with graph density, the average degree is more appropriate for indicating the group trust distribution. For example, although the density of syntactic graph ⑤ is roughly a half of the density in the graph ①, their group trust distributions are similar because they have roughly the same average degree.

Based on the group trust distributions, we also plot the probability $P(A_j|\Phi_{ij})$ with different Φ_{ij} in four settings of λ and β in Fig. A.19(a)–(f). These figures show the $P(A_j|\Phi_{ij})$ consistently decreases when Φ_{ij} increases. Moreover, the $P(A_j|\Phi_{ij})$ can reach a smaller value in the graph which has higher average degree and graph density.

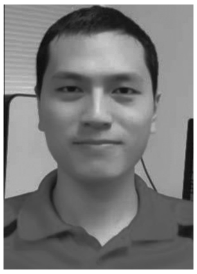
To investigate how graph properties (i.e., average degrees and graph densities) affect the global trust propagation, we show $P(A_j|L_i)$ in these six graphs in Fig. A.20(a)–(f). In these figures, the $P(A_j|\Phi_{ij} = 1)$ represents how likely v_j is an adversary if v_i has local trust in v_j. If $P(A_j|L_i) < P(A_j|\Phi_{ij} = 1)$, that means the people who receive global trust through a trust path consisting of L_i trust edges are less likely to be adversaries than the people who receive local trust. As a result, when SGor can achieve $P(A_j|L_i) < P(A_j|\Phi_{ij} = 1)$ using appropriate L_i and Φ_h, the trust propagation could not degrade the capability of resisting adversaries.

To maintain the capability of resisting adversaries during trust propagation, we summarize appropriate L_i according to different Φ_h in SGor for these six syntactic graphs in Table A.4. Apparently, the graph with larger average degree and higher density supports a larger L_i for trust propagation without reducing the capability of evading adversaries.

References

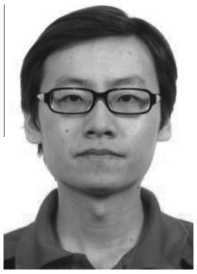
[1] Krishna P.N. Puttaswamy, Alessandra Sala, Ben Y. Zhao, Improving anonymity using social links, in: Proc. Workshop on Secure Network Protocols, 2008.
[2] Aaron Johnson, Paul Syverson, More anonymous onion routing through trust, in: Proc. Computer Security Foundations Symposium, 2009.
[3] Aaron Johnson, Paul Syverson, Roger Dingledine, Nick Mathewson, Trust-based anonymous communication: adversary models and routing algorithms, in: Proc. ACM Conference on Computer and Communications Security, 2011.
[4] Paul Syverson, Gene Tsudik, Michael Reed, Carl Landwehr, Towards an analysis of onion routing security, in: Proc. Privacy Enhancing Technologies Symposium, 2000.
[5] Matthew K. Wright, Micah Adler, Brian Neil Levine, Clay Shields, The predecessor attack: an analysis of a threat to anonymous communications systems, ACM Transactions on Information and System Security 7 (2004) 489–522.
[6] Steven J. Murdoch, George Danezis, Low-cost traffic analysis of Tor, in: Proc. IEEE Symposium on Security and Privacy, 2005.
[7] Lasse Øverlier, Paul Syverson, Locating hidden servers, in: Proc. IEEE Symposium on Security and Privacy, 2006.

[8] Kevin Bauer, Damon McCoy, Dirk Grunwald, Tadayoshi Kohno, Douglas Sicker, Low-resource routing attacks against Tor, in: Proc. ACM Workshop on Privacy in the Electronic Society, 2007.
[9] Nicholas Hopper, Eugene Y. Vasserman, Eric Chan-Tin, How much anonymity does network latency leak? in: Proc. ACM Conference on Computer and Communications Security, 2007.
[10] Zhen Ling, Junzhou Luo, Wei Yu, Xinwen Fu, Dong Xuan, Weijia Jia, A new cell counter based attack against Tor, in: Proc. ACM Conference on Computer and Communications Security, 2009.
[11] Nathan S. Evans, Roger Dingledine, Christian Grothoff, A practical congestion attack on Tor using long paths, in: Proc. USENIX Security Symposium, 2009.
[12] Ye Zhu, Xinwen Fu, Bryan Graham, Riccardo Bettati, Wei Zhao, Correlation-based traffic analysis attacks on anonymity networks, IEEE Transactions on Parallel and Distributed Systems 21 (2010) 954–967.
[13] Zhen Ling, Junzhou Luo, Wei Yu, Xinwen Fu, Weijia Jia, Wei Zhao, Protocol-level attacks against tor, Computer Networks 57 (2012) 869–886.
[14] Roger Dingledine, Nick Mathewson, Paul Syverson, Tor: the second-generation onion router, in: Proc. USENIX Security Symposium, 2004.
[15] Peng Zhou, Xiapu Luo, Rocky K.C. Chang, More anonymity through trust degree in trust-based onion routing, in: Proc. International Conference on Security and Privacy in Communication Networks, 2012.
[16] David Goldschlag, Michael Reed, Paul Syverson, Onion routing for anonymous and private Internet connections, Communications of the ACM 42 (1999) 39–41.
[17] George Danezis, Claudia Diaz, Carmela Troncoso, Ben Laurie, Drac: an architecture for anonymous low-volume communications, in: Proc. Privacy Enhancing Technologies Symposium, 2010.
[18] Prateek Mittal, Matthew Wright, Nikita Borisov, Pisces: Anonymous communication using social networks, in: Proc. Network and Distributed System Security Symposium, 2013.
[19] Bimal Viswanath, Alan Mislove, Meeyoung Cha, Krishna P. Gummadi, On the evolution of user interaction in Facebook, in: Proc. Workshop on Online Social Networks, 2009.
[20] Christo Wilson, Bryce Boe, Alessandra Sala, Krishna P.N. Puttaswamy, Ben Y. Zhao, User interactions in social networks and their implications, in: Proc. European Conference on Computer Systems, 2009.
[21] Zhen Ling, Junzhou Luo, Wei Yu, Xinwen Fu, Dong Xuan, Weijia Jia, A new cell-counting-based attack against Tor, IEEE/ACM Transactions on Networking 20 (4) (2012) 1245–1261.
[22] Amir Houmansadr, Nikita Borisov, SWIRL: a scalable watermark to detect correlated network flows, in: Proc. Network and Distributed System Security Symposium, 2011.
[23] Xiapu Luo, Junjie Zhang, Roberto Perdisci, Wenke Lee, On the secrecy of spread-spectrum flow watermarks, in: Proc. European Symposium on Research in Computer Security, 2010.
[24] Xiapu Luo, Peng Zhou, Junjie Zhang, Roberto Perdisci, Wenke Lee, Rocky K.C. Chang, Exposing invisible timing-based traffic watermarks with backlit, in: Proc. Annual Computer Security Applications Conference, 2011.
[25] John R. Douceur. The Sybil attack, in: Revised Papers from the First International Workshop on Peer-to-Peer Systems, 2002.
[26] Haifeng Yu. Sybil defenses via social networks: a tutorial and survey. in: SIGACT News, 42, pp. 80–101.
[27] Ralph Gross, Alessandro Acquisti, Information revelation and privacy in online social networks (the facebook case), in: Proc. Workshop on Privacy in the Electronic Society, 2005.
[28] Yabing Liu, Krishna P. Gummadi, Balachander Krishnamurthy, Alan Mislove, Analyzing facebook privacy settings: User expectations vs. reality, in: Proc. Internet Measurement Conference, 2011.
[29] Yazan Boshmaf, Ildar Muslukhov, Konstantin Beznosov, Matei Ripeanu, The socialbot network: When bots socialize for fame and money, in: Proc. Annual Computer Security Applications Conference, 2011.
[30] Alan Mislove, Bimal Viswanath, Krishna P. Gummadi, Peter Druschel, You are who you know: inferring user profiles in online social networks, in: Proc. ACM International Conference on Web Search and Data Mining, 2010.
[31] Ramanthan Guha, Ravi Kumar, Prabhakar Raghavan, Andrew Tomkins, Propagation of trust and distrust, in: Proc. International conference on World Wide Web, 2004.
[32] Dinh Nguyen Tran, Bonan Min, Jinyang Li, Lakshminarayanan Subramanian, Sybil-resilient online content rating, in: Proc. USENIX Symposium on Networked Systems Design and Implementation, 2009.
[33] Prateek Mittal, Matthew Caesar, Nikita Borisov, X-vine: Secure and pseudonymous routing in dhts using social networks, in: Proc. Network and Distributed System Security Symposium, 2012.
[34] Haifeng Yu, Phillip B. Gibbons, Michael Kaminsky, Feng Xiao, Sybillimit: a near-optimal social network defense against sybil attacks, IEEE/ACM Transactions on Networking 18 (2010) 885–898.
[35] NetworkX. <http://networkx.github.io/>.
[36] Albert-László Barabási, Réka Albert, Emergence of scaling in random networks, Science (1999).



Peng Zhou received the BSc and MSc degrees in Electrical Engineering from Donghua university in 2004 and 2007, respectively. He then joined Hewlett–Packard Company, Shanghai, China and worked as a firmware engineer and security software engineer for three years. He is now working toward the PhD degree in the Department of Computing at the Hong Kong Polytechnic University. His research interests cover network security and privacy.

Xiapu Luo received his Ph.D. degree in Computer Science from the Hong Kong Polytechnic University in 2007 and then spent two years at the Georgia Institute of Technology as a post-doctoral research fellow. He is now a research assistant professor in the Department of Computing at the Hong Kong Polytechnic University. He is also a researcher affiliated with the Shenzhen Research Institute of the Hong Kong Polytechnic University. His current research focuses on network security and privacy, Internet measurement, and Smartphone security.



Ang Chen is a first-year PhD student in the Computer and Information Science Department, University of Pennsylvania. His research interests primarily lie in distributed systems and networking, including Internet measurement, security and privacy issues of distributed systems, and others.

Rocky K.C. Chang received the PhD degree in computer engineering from Rensselaer Polytechnic Institute. Immediately after that, he joined the IBM Thomas J. Watson Research Center working on performance analysis and simulation tools. He then joined the Department of Computing at the Hong Kong Polytechnic University, where he is now an associate professor. He is leading an Internet Infrastructure and Security Laboratory, addressing problems in network security, network measurement, and network operations and management. He is a member of the IEEE and ACM.