Leveraging Social Feedback to Verify Online Identity Claims

MICHAEL SIRIVIANOS, Cyprus University of Technology
KYUNGBAEK KIM, Chonnam National University
JIAN WEI GAN, TellApart
XIAOWEI YANG, Duke University

Anonymity is one of the main virtues of the Internet, as it protects privacy and enables users to express opinions more freely. However, anonymity hinders the assessment of the veracity of assertions that online users make about their identity attributes, such as age or profession. We propose FaceTrust, a system that uses online social networks to provide lightweight identity credentials while preserving a user's anonymity. FaceTrust employs a "game with a purpose" design to elicit the opinions of the friends of a user about the user's self-claimed identity attributes, and uses attack-resistant trust inference to assign veracity scores to identity attribute assertions. FaceTrust provides credentials, which a user can use to corroborate his assertions. We evaluate our proposal using a live Facebook deployment and simulations on a crawled social graph. The results show that our veracity scores are strongly correlated with the ground truth, even when dishonest users make up a large fraction of the social network and employ the Sybil attack.

Categories and Subject Descriptors: C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed applications*; H.3.5 [**Information Storage and Retrieval**]: Online Information Services—*Web-based services*; H.1.2 [**Models and Principles**]: User/Machine Systems—*Human factors*

General Terms: Algorithms, Design, Security

Additional Key Words and Phrases: Anonymity, online credential, online identity, online social network, privacy

ACM Reference Format:
Sirivianos, M., Kim, K., Gan, J. W., and Yang, X. 2014. Leveraging social feedback to verify online identity claims. *ACM Trans. Web* 8, 2, Article 9 (March 2014), 38 pages.
DOI:http://dx.doi.org/10.1145/2543711

1. INTRODUCTION

Rich social interactions take place on the Web, such as blogging, shopping, chatting, working, and playing. However, unlike social interactions in the physical world, the Web has largely hidden the identity of online users. "On the Internet, nobody knows you are a dog," says the famous Peter Steiner cartoon. Anonymity has brought much benefit, such as enabling users to express opinions freely. However, it makes what and

This work was funded in part by NSF CAREER Award CNS-0845858 and Award CNS-0925472. This research was supported by the MSIP (Ministry of Science, ICT & Future Planning), Korea, under the ITRC (Information Technology Research Center) support program (NIPA-2013-H0301-13-3005) supervised by the NIPA (National IT Industry Promotion Agency).
This article is an extended version of the earlier publication, Assessing the Veracity of Identity Assertions via OSNs, COMSNETS 2012.
Authors' addresses: M. Sirivianos, Cyprus University of Technology, Cyprus; K. Kim (corresponding author), Chonnam National University, Korea; email: kyungbaekkim@jnu.ac.kr; J. W. Gan, TellApart, CA; X. Yang, Duke University, NC.

who to believe online challenging. Individuals that hide their real identity attributes may defraud naive users.

Consider this real-life example. Alice is shopping for a food scale and she finds a rave review “Worth DOUBLE the Money” [Merkin 2006] from a user claiming “I was a chef for many years”. Should she believe this review, given that she is aware of authors or users with vested interests in a company who have been caught creating fake positive reviews for their own books [Harmon 2004] or the company’s products [Parsa 2009]? Real-world remedies to this problem typically forgo a user’s anonymity. For instance, Amazon provides a “Real Name” badge to a user that wishes to sign his posts by his real name. Amazon verifies a user’s name using his credit card information. Moreover, verifying a user’s identity by examining official certificates or meeting in person can be costly and time consuming.

This situation prompts the question: can a user cost-effectively establish online identity assertion veracity without sacrificing his anonymity? One approach is to use personal digital certificates issued by a trusted Certificate Authority (e.g., VeriSign), and to apply techniques such as *idemix* [Camenisch and van Herreweghen 2002] to make the certificates anonymous, unlinkable, and nontransferable. However, this approach involves centralized manual verification. Thus it could be a financial and a usability burden on users [Whitten and Tygar 1999].

In this article we propose FaceTrust, which is a system that enables online personas to inexpensively obtain credentials that indicate the veracity of their identity statements without sacrificing their anonymity. The insight of FaceTrust is that online users or services do not require strong authentication in many settings, and can benefit greatly from likely-to-be-true identity information. For example, a user who contemplates purchasing a book may benefit by knowing that a reviewer’s declared profession is more likely to be true than another reviewer’s. Similarly, it may suffice for an adult site to know that a user’s age information is likely to be true. In the adult site setting in particular, the currently deployed solution relies solely on Web sites asking the user to state, under the penalty of perjury, whether he is of the proper age. This solution does nothing but transfer all the responsibility for a potential violation to the lying user. The Web sites themselves are not required to pose any additional obstacles to the users. Our solution offers regulators the option to require online services to raise the bar for violators, while preserving user anonymity. Our veracity scores are manipulation-resistant trust values that can assist verifiers to make more informed decisions on whether to accept an identity claim.

FaceTrust mines and enriches information embedded in Online Social Networks (OSNs) to provide lightweight and flexible digital credentials of the identity assertions. Leveraging the fact that OSNs already allow users to express a limited form of trust relationships by using friend links, FaceTrust extends this property by allowing users to declare whether they consider the identity assertions of their friends credible (Section 3.1). In particular, a user who wishes to obtain a credential posts short assertions about himself on his OSN profile in the form of a poll, such as “Am I really over 18?” Because the identity information on the OSN profile is inserted by the user, the OSN provider cannot directly infer its veracity. Thus, the OSN enables the user to ask his friends to provide a new type of social feedback, that is, to respond to this poll by tagging the user’s asserion as `true` or `false`. Based on the tagging information, the OSN employs a veracity scoring mechanism (Section 3.3.2) to estimate a score that reflects how credible an assertion is. We call this score *assertion veracity* (Section 3.2).

The assertion veracity scoring mechanism also employs transitive trust inference [Guha et al. 2004], which needs to be attack resistant because users may post false assertions, tag incorrectly or lie, and create Sybil accounts [Douceur 2002]. The intuition behind the proposed trust inference scheme is that benign (honest) users tend to tag correctly and similarly. A user's tags are compared with those from his friends on the set of assertions they both tag, and the similarity between the tags is considered as pairwise trust values on the social graph edges. Based on the similarity-annotated social graph, a Sybil-resistant *tagger trustworthiness* score for each user is computed. For this purpose, a new max-flow-based scoring mechanism is proposed, which is more scalable than existing approaches. The proposed scoring mechanism seeds trust at preselected known honest users and propagates it along the similarity-annotated edges, resulting in dishonest users that tag falsely to have substantially lower trust than honest users. Finally, an assertion's veracity is derived by combining its tags weighted by their tagger's trustworthiness.

After deriving an assertion's veracity, the OSN issues a credential in the form of `{assertion, veracity, content, context}` (Section 3.4). Verifiers (online services or human users) can use this OSN-issued credential to regulate their interactions with the user that posted the assertion. Users and applications can combine the assertion veracity value with other contextual information to make more informed decisions during online interactions. For example, in order for a user to decide whether a computer science book on Amazon is worth buying, he may consider both the fact that a positive review is written by a persona certified by FaceTrust to be a CS professor, and the fact that the review is well articulated. For usability, our context-specific credential scheme allows a user to certify his online assertions with a simple Web interface without involving user-side cryptography.

To evaluate FaceTrust, we have built and deployed a FaceTrust application which has amassed over 1000 users. We have also evaluated FaceTrust through a series of extensive simulations on a 200K-user sample of a crawled social graph (Section 6). The results of our evaluation show that FaceTrust assigns high veracity to true assertions and low veracity to false ones, even when a large fraction of the network is dishonest and employs the Sybil attack.

The rest of the article is organized as follows: Section 2 describes the overview of FaceTrust with an example and discusses FaceTrust's assumptions and goals. Section 3 provides the design of FaceTrust in detail and the analysis of attack resistance of FaceTrust is discussed in Section 4. Section 5 describes the implementation of FaceTrust and Section 6 presents an evaluation of FaceTrust. Section 7 discusses related work and Section 8 concludes.

2. OVERVIEW

FaceTrust involves the following three roles (Figure 1): (a) the OSN provider that maintains the social network and its users' profiles, and performs trust computations; (b) online users that maintain accounts with the OSN and wish to present OSN-issued credentials; and (c) credential verifying online services or users.

2.1. An Example

Figure 1 illustrates an age verification example to shed light on how FaceTrust roles interact. User u attempts to access an age-restricted movie at the Netflix Web site. At the same time, u does not wish to reveal his real identity to Netflix. With FaceTrust, Netflix can demand an OSN-issued age credential from the user to allow access to its content.

To obtain this credential, the user u must have posted an age assertion on his OSN profile, and requested his friends to tag the veracity of his age assertion before he attempts to access the age-restricted content. In this example, user u has asserted that his age is 21, and three of his friends, users x, y, and z, have tagged the assertion with boolean values `true`, `true`, and `false` respectively. Because not all users are equally trustworthy, the OSN provider has computed a trustworthiness score (w) for each

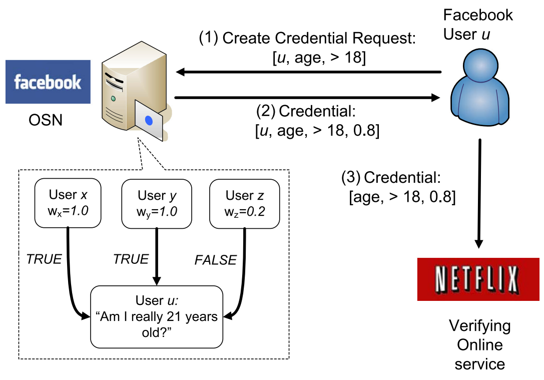

Facebook User u
(1) Create Credential Request: [u, age, > 18]
facebook
OSN
(2) Credential: [u, age, > 18, 0.8]
(3) Credential: [age, > 18, 0.8]
User x w_x=1.0
User y w_y=1.0
User z w_z=0.2
TRUE
TRUE
FALSE
User u: "Am I really 21 years old?"
NETFLIX
Verifying Online service

Fig. 1. FaceTrust overview and an age verification example.

tagger x, y, and z by analyzing the social graph and their tagging history as described in Section 3.3. The OSN provider computes an overall veracity score for user u's age assertion (0.8 in this example) by aggregating u's friends' tagged values weighted by their trustworthiness scores (Section 3.2).

Subsequently, the OSN issues an age credential with a veracity score that certifies that the user belongs to the restricted age group. The user presents this credential to Netflix to access its content (e.g., by submitting the credential URL, or the credential ID). For ease of use, instead of using cryptographic digital certificates, FaceTrust implements Web-based identity credentials using an XML API for online services such as Netflix, or through a simple Web page for human users (Section 3.4).

2.2. More Motivating Examples

In addition to age verification, FaceTrust credentials will benefit Internet users and online services in many other ways.

Assessing the authority or relevance of online reviews or ratings with profession credentials. Many Internet users read online reviews before making purchase decisions. Intuitively, expert opinions of an online product may appear more authoritative to many readers. For instance, a reader may place more weight on a review for a networking textbook by a computer science professor than by an average user. With FaceTrust, if an expert user desires to appear more authoritative, he may request a profession credential from his OSN provider and include it with his review.

Note that the goal of FaceTrust is not to explicitly assess the correctness of the review. Instead, FaceTrust only aims at verifying the identity attribute (e.g., profession) of the user that posted the review. This is because FaceTrust is suitable for assessing the veracity of ground-truth statements (ones that are universally accepted as `true` or `false`) and not for assessing statements that are subject to personal taste or opinion.

Verifying participant eligibility. A citizen journalism site may wish to verify that a user actually resides in a specified area before it accepts his report on an event that took place there. Similarly, Wikipedia, online auction sites, online statistical surveys, and online dating sites [Norcie et al. 2013] may wish to restrict participants to certain groups of people, such as people with particular expertise, residents in a certain geographic area, or people of a certain age range. FaceTrust can assist legitimate participants to obtain credentials that certify their eligibility.

Preventing online fraud. Scammers commonly respond to online postings alleging to be prospective participants in legitimate transactions (e.g., a potential tenant of an apartment), and aiming to commit "advance-fee" fraud [Rental-Scam 2013]. Such attacks could possibly be averted if online users had a way to infer when scammers lie about their location, affiliation, or age. To this end, a classifieds service such as Craigslist or its users could use FaceTrust to verify identity attributes of users that post or respond to ads.

2.3. Assumptions

In designing FaceTrust, we make the following assumptions.

Users carefully vet FaceTrust friend requests. This assumption is based on the fact that establishing friend connections in the FaceTrust social network is a resource-intensive task. We assume that Sybils establish a limited number of attack edges due to the difficulty of soliciting and maintaining reciprocal social relationships [Cao et al. 2012; Yu et al. 2006]. Although previous studies [Bilge et al. 2009; Boshmaf et al. 2011] suggest that fake identities can befriend others, a recent study shows that most of their connections are established with other fakes [Motoyama et al. 2011]. Users can use common-secret-based techniques to verify that an OSN friend request originates from a real acquaintance and not an imposter [Baden et al. 2009]. This assumption also implies that a user selects as OSN friends users that will not try to harm him by tagging his honest assertions as `false`. Therefore, the FaceTrust social network is a more carefully vetted subgraph of the OSN graph. Examples of real-world social graphs that consist of mostly carefully vetted edges are the Google+ and Facebook subgraphs that contain only the edges between users that appear in mutually close circles and friend lists, respectively.

OSN provider as a trusted credential issuing authority. We assume that the OSN provider reliably issues credentials based on the input of its users. Also, the OSN provider does not reveal a user's tags to others, otherwise it would lose the trust of its users. Furthermore, users may wish to remain anonymous and untraceable by the verifiers. We make the explicit assumption that the OSN (or FaceTrust) provider is bound by end-user agreements to protect the privacy of its users by not revealing their identity and the list of online services or users that verify its users' credentials. We assume that unless the user himself includes personally identifiable information in his credentials, this information will not leak to the verifing users who view the credential. For example, in the case of the Amazon review, the readers of the review would know that the user is a CS professor but they would not know what his exact rank is and in which university he works. In the case of an age-restricted site, the site would learn only that the user is likely to be over 18 and it would not have access to any other item of the user's profile.

We note that it is common for users to rely on trusted service providers for critical issues regarding their security, for example, using certificate authorities for the issuance of certified key pairs or entrusting private information to Web mail services. Trust in centralized and specialized online security service providers is a common occurrence and is part of several viable business models.

Trustworthy users tag mostly correctly, as well as post true identity assertions. When trustworthy users (*honest*) tag a same identity assertion their tags mostly match, because an assertion in FaceTrust is about ground truth (e.g., age, profession, sex, etc.) rather than personal taste. This assumption is validated to some extent in Section 6.2. The users that consistently tag mistakenly, not due to malicious intent, but due to lack of knowledge, are treated as *dishonest* (Section 2.4).

Transitivity of trust. We assume the transitivity of trust in the online social network setting. Transitivity is a highly desired property that leads to higher accuracy for trust metrics, especially in a system where global trust information is lacking [Richters and Peixoto 2011]. In particular, Google's PageRank [Page et al. 1999] and its Web-spam-resilient variants, such as TrustRank [Gyöngyi et al. 2004], rely on the transitivity of the trust that is expressed as directional hyperlinks between Web pages. In addition, highly predictive blacklisting leverages the transitivity of trust between email spam detectors, which in that particular case manifests itself as similarity between attack reports [Zhang et al. 2008]. Furthermore, SybilRank [Cao et al. 2012] is a recent system that relies on trust transitivity and is successfully deployed to uncover Sybils in a real OSN. In that case, trust is expressed as social links between honest accounts. Also, the transitivity of trust is widely used in the X.509 public key infrastructure scheme's chain of trust [Perlman 1999; Wikipedia 2013].

Trust may not be sufficiently transitive in some contexts. For example, the study in Ramasubramanian and Sirer [2005] uncovers several security vulnerabilities of the domain name system that stem from the system's reliance on trust transitivity. In addition, an authorized digital certificate for the "*.google.com" domain was issued due to a compromised chain of trust [Langley 2013]. We do not claim to have a solution that accurately measures the trust between users, but a system that leverages trust transitivity to the degree it actually manifests itself. In this article, FaceTrust transforms the social network graph into a trust graph by annotating edges with a value that is derived from a combination of the similarity of tags between users and a trust value that users explicitly assign to each other (Section 3.3). This design combines the transitivity of trust along social edges [Cao et al. 2012] with the transitivity of the similarity of feedback [Zhang et al. 2008].

2.4. Threat Model

FaceTrust's design copes with the following types of malicious users that aim to subvert the system.

Dishonest assertion posters and taggers. We consider dishonest users that are primarily interested in posting dishonest assertions to misrepresent their identities. These dishonest users can collude with other dishonest users that tag their false assertions as `true`.

Sybil taggers. Dishonest users can launch the Sybil attack [Douceur 2002] by creating many fake accounts under their control. A dishonest user that creates Sybils can employ them to tag the false assertions of the creator's colluders as `true`. Sybils may also infiltrate social network graphs [Irani et al. 2011; Yang et al. 2011] by befriending honest users or by being connected to dishonest users that are well connected to honest ones.

Sybil assertion posters. Dishonest users can create Sybils who are connected to them and post false assertions. These assertions are tagged by their creators and their colluders as `true`. This attack creates Sybil users that are not friends with honest users, thus their assertions are never tagged `false` by them. Consequently, it is easier for those Sybils to make their assertions have a relatively high veracity score.

Camouflage attack. This threat resembles what Kamvar et al. [2003] refer to as "malicious nodes with camouflage." Malicious nodes may initially behave honestly to accumulate trust for themselves or their friends, and they later attempt to defeat the system. One manifestation of this attack is the *tagger camouflage attack*. Dishonest users attempt to build up trust with honest users by always tagging similarly to the veracity that is currently displayed for the assertion. After they earn enough tagger trustworthiness, they tag dishonestly only for specific questions. The tagger camouflage attack also implies the case in which a user is not always tagging falsely, but instead has a mixed strategic behavior. Another manifestation of the camouflage attack is the *assertion poster camouflage attack*. A dishonest user posts several honest assertions. Both his honest and dishonest friends tag those assertions as `true`. As a result, if his dishonest and honest friends are also friends with each other, his dishonest friends build up trust with his honest friends. Consequently the dishonest friends' tagger trustworthiness increases.

Rational dishonest users. We assume that dishonest users who post false assertions and tag dishonestly are rational. Dishonest users benefit by obtaining a credential that makes a false assertion appear `true`. At the same time, they incur a cost every time they create a Sybil account, that is, the time needed to solve CAPTCHAs at registration time. They also incur a cost every time they coordinate with other dishonest users on which assertions to tag as `true`.

2.5. Goals

FaceTrust's design is driven by the following goals.

Attack resistant. Our system aims at making it difficult for false assertions to appear trustworthy by having high veracity. The system should be resilient to errors made by benign users and to manipulation by dishonest users. Although our design is attack mitigating, it cannot ensure the correctness of the veracity scores in the presence of devoted adversaries. Therefore, our system is not meant for guarding critical resources and the veracity scoring is relaxed to be within the range $[0, 1]$ rather than a binary `true` or `false` value. FaceTrust credentials alone cannot guarantee the truthfulness of a statement. For this reason we refer to our credentials as *relaxed*.

Informative. FaceTrust should offer users additional information on the veracity of identity assertions. Because it is relatively easy for a user who posts a truthful assertion to find friends who vouch for him, if an assertion has low veracity, users should reject it from the outset. If the assertion has high veracity, users should still employ other common-sense verification mechanisms or be aware that they are taking a risk by accepting the statement as `true`. The veracity of user assertions should correlate positively with the ground truth. When an assertion has higher veracity than another, this should indicate that it is more likely to be credible. Verifiers can define their own thresholds (possibly suggested by FaceTrust's admins) to map a veracity score to an action based on the application scenarios and their own risk tolerance. To ease the interpretation of the veracity score, it is desirable that the veracity of true and false assertions is greater or lower than 0.5, respectively. Users can use this measure by setting thresholds that we can suggest. For example, through experimentation in synthetic traces, honest assertions almost always get higher than 50% veracity, while dishonest ones get less than 30%.

Lightweight. We aim to provide credible identity information for online personas without centralized manual identity verification.

Flexible. Users should be able to obtain credentials on a variety of attributes, such as age, profession, etc. Users should also be able to conveniently obtain new credentials when their attributes change.

Practical. The system should be easy to use. It should not require users to deal with cryptographic primitives and shared secrets. It should require minimal upgrades of client software.

Secure. The credentials should satisfy authentication, that is, the verifier should be assured that a credential is issued by a trusted authority. They should satisfy integrity,

Table I. Key Notations

Name	Meaning
A_i^t	Assertion of type t posted by user j
w_j^t	Tagger trustworthiness of user j for type t
d_{ji}^A	Tag by friend j of user i on i's assertion A
a_A	Veracity of assertion A
ts_{ij}^t	Tagging similarity between users i and j for assertions of type t
N_t	Number of tags on common assertions of type t between users i and j
C_t	Number of equal tags on common assertions of type t between users i and j
hs_{ij}	History-defined similarity between users i and j for assertions of type t
us_{ij}^t	User-defined similarity between users i and j for assertions of type t
$a(N_t)$	logistic function of N_t used in combining user-defined similarity with tagging similarity
$G(V,E_t)$	Similarity-based trust graph for type t
S	Set of trusted sources (seeds)
$C_{supersource}$	Capacity at the supersource
p_d	Portion of the social graph that is dishonest

namely the assertion and context fields should be inalterable once the credential is issued, and the veracity field reflects the correctly computed assertion veracity score (Section 3.2). This guarantees that a user cannot forge the veracity score of his assertions and that a user cannot use somebody else's assertions to verify his identity attributes. Moreover, the credentials should preserve anonymity, that is, a user should be able to present credentials with no personally identifiable information.

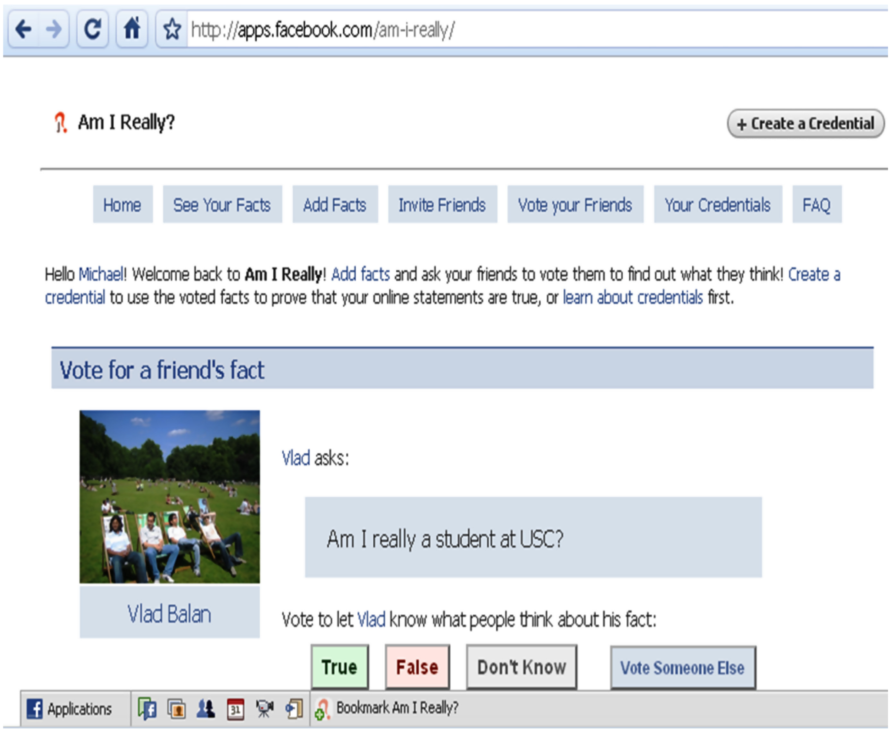
3. DESIGN

In this section the detailed design of FaceTrust is presented. Table I lists the key notations used in the article.

3.1. Social Tagging

FaceTrust elicits from its users a new type of feedback which it uses to assess the credibility of identity information. We refer to this feedback as "social tagging". It involves OSN users posting assertions about the attributes of their identities and their friends tagging them as `true` or `false`. Our approach is similar to the PGP Web of Trust [Zimmerman 2009], in that we allow only friends of a user to tag (certify) the user's assertion. Our rationale is twofold. First, most of the assertions posted by a user can only be reliably evaluated by people who know him (friends). Second, because a user has carefully vetted his friends, those friends are likely not to attempt to harm him by tagging his true identity assertions as `false`.

FaceTrust categorizes identity attribute assertions into various types such as age, address, profession, expertise, etc. For instance, for the type age, an assertion has the format `[{<,=,>}, number]`, for example, `[> 18]` means that the user claims to be older than 18. For the type location, the assertion has the format `[{country, state, city ...}, string]`, such as [country, US]. We use distinct types because a user's tendency to correctly tag assertions may vary by type (Section 3.2), and to address the camouflage attack (Section 3.3.1).



Hello Michael! Welcome back to **Am I Really**! Add facts and ask your friends to vote them to find out what they think! Create a credential to use the voted facts to prove that your online statements are true, or learn about credentials first.

Vote for a friend's fact

Vlad asks:

Am I really a student at USC?

Vlad Balan

Vote to let Vlad know what people think about his fact:

Fig. 2. Illustration of social tagging using the “Am I Really?” Facebook application.

For an assertion A_i^t of type t posted by a user i, i’s friend j may tag it as d_{ji}^A. Specifically, d_{ji}^A takes two values: `true` indicates that j believes i’s assertion, and `false` that it does not. A posted assertion and its associated tags are valid for a period of time set by the OSN provider depending on the assertion type. An assertion is uniquely identified by its {`type`, `assertion`} pair. A user cannot repost the same assertion and reset unfavorable tags before the assertion expires. Because the tags represent sensitive information, they are only known to the OSN provider and their tagger. In addition, the OSN provider does not reveal to the assertion poster the veracity of an assertion unless the assertion has accumulated a threshold number of tags to protect taggers’ privacy.

This design assumes that users are willing to tag their friends. This is a reasonable assumption because abundant evidence suggests that users may adopt social tagging. For example, “Friend Facts” is a popular application that presents a user with questions about his friends and asks him to vote to let them know what he thinks about them. It has amassed $\sim$ 4.5 million monthly active users. We further validate this assumption in Section 6.2 using data from our real-world deployment of the “Am I Really?” Facebook application: http://apps.facebook.com/am-i-really (Section 5). By using AIR, users post assertions on their OSN profile and tag their friends’ assertions. AIR employs a “game with a purpose” design to incentivize social tagging. A snapshot of social tagging using our Facebook application can be seen in Figure 2.

3.2. Assertion Veracity

A main challenge in FaceTrust’s design is to assess the veracity of user assertions. This task is difficult because dishonest users may post false assertions and strategize

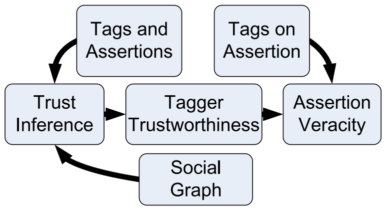


Fig. 3. Combining social tagging with trust inference to derive the veracity of user assertions.

to assign them high veracity, and benign users may make mistakes. To make this task tractable, we resort to providing a relaxed credential that binds an assertion to a veracity score between 0 and 1.

Definition 3.1. The *assertion veracity* of an assertion A_i^t is a score $0 \leq a_A \leq 1$ reflecting the truthfulness of an identity assertion A. It strongly and positively correlates with the truth, that is, an assertion with higher veracity than another is more likely to be true.

As shown in Figure 3, the inputs for computing an assertion's veracity are the tags on an assertion and the taggers' trustworthiness. A tagger j's trustworthiness w_j^t is a measure that estimates the trustworthiness of j's tags on assertions of type t. We compute this measure using the trust inference technique described in Section 3.3.2. We then weigh the tags of an assertion with their taggers' trustworthiness to score the assertion's veracity. Let F_i denote the set of friends of user i that have tagged the assertion A_i^t. To compute the veracity score a_A of A_i^t , the OSN provider aggregates the tags d_{ji}^A by i's friends as follows.

$$a_A = \max(\sum_{j \in F_i} w_j^t \cdot d_{ji}^A / \sum_{j \in F_i} w_j^t,\ 0) \tag{1}$$

We make the scoring of the veracity of an assertion conservative by assigning -1 to `false` tags, 1 to `true` tags, and normalizing negative veracity scores to zero. For instance, if an assertion has two tags `true` and `false` from two equally trustworthy taggers, its assertion veracity will be 0, not 0.5. Eq. (1) ensures that the sum of the weights w_j^t of `true` tags should be more than 0.75 of the sum of the weights of all users in F_i for a_A to have more than 0.5 veracity. Thus, this design is biased towards making it difficult for users to make false assertions have a high veracity score. However, malicious users may abuse this design to make `true` assertions of a user noncredible. We are tackling this attack by allowing only a user's friends to tag his assertions.

We use the additional condition that if the sum of the trustworthiness of the taggers of the assertion A_i^t is below a specified threshold M, a_A is 0. M can be proportional to the mean tagger trustworthiness of users. We use this condition to discount assertions that have been tagged only by a few users with low tagger trustworthiness.

$$a_A = 0 \quad \text{if} \quad \textstyle\sum_{j \in F_i} w_j^t < M \tag{2}$$

We analyze the assurances provided by the assertion veracity mechanism in Section 4.1.

3.3. Tagger Trustworthiness

Definition 3.2. The *tagger trustworthiness* of a user j is the integer score $0 \leq w_j^t \leq T_{max}$ that indicates whether a tagger j is honest or correct in his assessments of the veracity of assertions of a specific type. This score strongly and positively correlates with the ground truth, that is, a tag by a user with higher trustworthiness is more likely to correspond to the reality.

How can FaceTrust reliably determine a tagger's trustworthiness w_j^t? To address this problem we resort to a Sybil-resistant *trust inference* technique. A trust inference algorithm refers to the process of computing the trustworthiness of a node in a graph by exploiting the transitivity of trust. The algorithm assumes that a few selected nodes in the graph are fully trustworthy (trust seeds). It then analyzes the trust graph to determine how trust propagates to other nodes.

We face two challenges in determining the tagger trustworthiness w^t. First, trust inference uses a *trust graph*, where an edge between two users i and j is explicitly labeled with the degree of trust that i places on j. However, this explicit trust information is not available in a social network graph. Second, how should we compute the tagger trustworthiness w^t, given that different trust inference algorithms exist and each has its own strengths? We describe how we address each challenge in turn in Section 3.3.1 and Section 3.3.2.

3.3.1. Tagging Similarity. We address the first challenge by using tagging similarity between two friends to approximate explicit trust. Recall that our assumption is that honest users tend to tag correctly and similarly, and note that tagging similarity is transitive. The tagging similarity ts_{ij}^t between two friends i and j for an assertion type t is computed from two sources: a history-defined similarity hs_{ij}^t and a user-defined similarity us_{ij}^t.

We compute the history-defined similarity between two friends using a formula that resembles the Jaccard index [Jaccard 1901]. Let N_t be the total number of assertions of type t that friends i and j both have tagged. Let C_t be the number of tags on the set of common assertions for which i and j are in agreement. The history similarity hs_{ij}^t between i and j for type t is computed as $hs_{ij}^t = C_t/N_t$. If $N_t = 0$, the similarity is equal to 0.

We incorporate user-defined similarity for the case when the number of tags is insufficient to calculate a meaningful history-defined similarity. The user-defined similarity us_{ij}^t reflects whether a user i trusts that the friend j will honestly tag assertions of a type t. To acquire the user-defined similarity, we use special assertions for each type t - "Do I honestly tag the <type> assertions of my friends?". Each user j posts these assertions on his OSN profile. If j's friend i tags it as `true`, us_{ij}^t equals 1; otherwise, it is 0.

We combine user-defined similarity with history-defined similarity to obtain the final tagging similarity between two friends in the OSN social graph: $ts_{ij}^t \leftarrow a \cdot hs_{ij}^t + (1 - a)us_{ij}^t$, where $0 \leq a \leq 1$. We vary the parameter a depending on how many common assertions N_t of the same type t users i and j have tagged; the larger N_t is, the higher a should be. When N_t is large, we presume that hs_{ij}^t approximates the likelihood that two friends would tag an assertion with the same value in the future more accurately than a manually specified value us_{ij}^t. However, when N_t is small, we use the user-defined value us_{ij}^t to approximate this likelihood. The parameter a is computed using the logistic function: $a = (1 + e^{b-N_t})^{-1}$. b is a small constant, and we set $b = 5$ in this article. The logistic function is S-shaped. For example, if we set $b = 5$, for $N_t \leq 2$, $a(N_t)$ would be less than 0.05. However, when N_t exceeds the threshold 3, a increases drastically until it becomes 0.5 for $N_t = 5$. For $N_t = 10$, $a(N_t)$ approximates 1.0.

We then transform the social graph into a trust graph by assigning the tagging similarity ts^t_{ij} to be the weight of a trust graph edge from a friend i to a friend j. We refer to this augmented graph as the similarity-based trust graph $G(V, E_t)$. Note that this is a directed graph, as the user-defined similarity us_{ij} is directional.

We have a distinct similarity-based trust graph for each type of assertion to mitigate camouflage attacks (Section 2.4). Due to this design an attacker is forced to tag honestly many assertions of the same type in order to boost its tagger trustworthiness. As a result, he is less flexible in his choice of which assertions to tag and how.

We compute tagging similarity only between friends to constrain the edges in the trust graph to be the same as the ones in the social graph. This design choice is critical in making our assertion veracity scoring algorithm attack resilient, because establishing social edges is a resource-intensive task [Yu et al. 2008] (Section 2.3). FaceTrust's Sybil defense relies on the fact that malicious users can establish a limited number of trust relationships with real humans (Section 2.3) [Levien 2003; Yu et al. 2008]. However, it raises the concern that if two friends do not share common friends, they may not have enough common assertions to tag. Fortunately, OSN measurement studies [Ahn et al. 2007; Mislove et al. 2007] show that the clustering coefficient in social graphs is one to five orders of magnitude higher than in Erdös-Rényi random graphs and preferential-attachment-constructed random power-law graphs. This result implies that two friends of a single user are more likely to be friends as well.

3.3.2. MaxTrust : Max-Flow-Based Trust Inference. Once we have converted a social graph into a trust graph, the challenge lies in computing a tagger's trustworthiness. To this end, we consider the max-flow-based broad class of trust inference algorithms [Cheng and Friedman 2005; Levien 2003; Levien and Aiken 1997; Reiter and Stubblebine 1999; Tran et al. 2009]. It has been shown that max-flow-based trust inference is more resilient to attackers because it considers multiple trust paths [Cheng and Friedman 2005; Levien and Aiken 1997; Reiter and Stubblebine 1997]. This is in contrast to trust metrics considered in other systems (e.g., the maximum trust path used in Credence [Walsh and Sirer 2006]). It has also been shown that a group max-flow-based trust metric [Levien 2003] is sum-Sybilproof, that is, an attacker cannot substantially increase the sum of the trust values of users under his control by introducing many Sybils.

The common element among trust inference methods is that trust flows from a few select trust seed users (trust seeds) and propagates to the other users in the trust graph. A seed is a highly trusted user, such as a trusted employee of the OSN provider that also verifies and tags assertions of many of his acquaintances. The specifics of the trust inference method determine how trust propagates in the graph. Our trust inference scheme should assign high trust to users that are well connected with the trust seeds and vote similarly to them. It should also assign lower trust to dishonest users that happen to be well connected but vote dissimilarly to the trust seeds. Finally, it should assign low trust to Sybil users that are often connected only to their dishonest creator users.

What renders a trust inference method Sybil resilient is the *bottleneck property* [Levien 2003], which we define as follows: "the trust that flows to the region of the graph that consists of dishonest users and their Sybils is limited by the edges connecting the dishonest region with the region that consists of trust seeds and honest users."

In addition, the selection of the trust seeds and the number of trust seeds is paramount to the attack resilience of the system. This is because an attacker that manages to befriend trust seeds and to build up high tagging similarity with them can greatly manipulate trust assignment. When the trust inference method employs numerous trust seeds, a dishonest user would need to identify and target many of them in order to be effective. Note that the complete trust graph itself is not made public, therefore locating a trust seed can be a difficult task for attackers.

Nevertheless, it is possible for dishonest users to infer a portion of the topology and identify a trust seed. Hence, one desirable feature of trust inference methods is to be efficiently computable for numerous trust seeds. To this end, the method's computation cost should be mostly independent of the number of trust seeds. One of our contributions is a max-flow-based trust inference method, called *MaxTrust*, for computing the tagger trustworthiness w_j^t with this desirable feature. We analyze the assurances provided by this mechanism in Section 4.2.

We also note that determining which users in a social graph can be designated as trust seeds is a challenge. Gyöngyi et al. [2004] addressed this challenge in the context of TrustRank, an eigenvector-based trust inference method for Web pages. TrustRank uses a PageRank-based algorithm to select seeds, and the set of selected seeds are verified by human experts prior to being used by the TrustRank algorithm. We apply the same solution to select MaxTrust seeds. More recently, Wu et al. proposed improvements over the seed selection algorithm [Wu et al. 2006] introducing the topical TrustRank.

Advogato and MaxTrust. Our method is inspired by the Advogato [Levien 2003] trust metric. Both Advogato and MaxTrust satisfy the *bottleneck property*. In particular, assuming that Sybils are only connected to their dishonest user creator, they ensure that the sum of the tagger trustworthiness of the creator and its Sybils does not exceed the sum of the capacity of the creator's incoming edges in the similarity-based flow graph.

Advogato determines the set of users that can be trusted by at least a certain level w on a trust graph, where a directed edge $u \rightarrow v$ indicates that user u trusts v by at least w. Advogato transforms the trust inference problem into a problem of maximum flow from a *single trust seed user* to a virtual supersink user. The capacity of the users in the flow graph is distributed such that the sum of the capacity of users at the same shortest hop distance from the trust seed is approximately equal to the capacity of the seed.

It splits each user into two virtual users (+ and −) and draws an additional edge of capacity 1 from the + virtual user to the supersink. The capacity of the edge connecting the + to the − virtual trust seed user is approximately the number of users in the trust graph that are expected to be trusted by at least x. The sum of the capacity of the $+\rightarrow-$ edges at each shortest hop distance from the trust seed is approximately equal to the capacity of the trust seed $+\rightarrow-$ edge. A user is considered trusted by at least w if the maximum flow solution has flow 1 on his $+\rightarrow$supersink edge.

Because Advogato chooses a single trust seed as the source of its max-flow computation, a dishonest user that is close to the seed can have high capacity. As a result, it can have many of its Sybils accepted at the same trust level as him. To mitigate this problem, one has to use multiple trust seeds from a set $S \subset V$, run the Advogato max-flow computation $|S|$ times, and average the resulting trust value across the runs. Compared to Advogato, MaxTrust's main advantage is that it does not need to be run for each trust seed. Instead in a single run (max-flow computation), it considers all the trust seed users. This results in MaxTrust being $\Theta(|S|)$ times more efficient than Advogato.

To assign tagger trustworthiness to a user using Advogato we need to run a max-flow computation for each nonzero trust level w, pruning at each run the edges that correspond to pairwise trust less than w. MaxTrust computes the tagger trustworthiness $0 \leq w_i^t \leq T_{max}$ in one run, but the optimal (nonheuristic) max-flow computation is

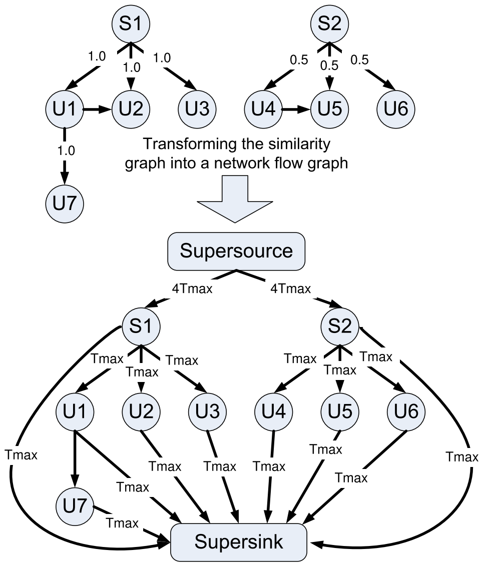


Fig. 4. The tagging similarity-based trust graph and its conversion into a MaxTrust network flow graph. The capacity $C_{supersource}$ in this example is $8 \cdot T_{max}$. MaxTrust results in all users except $U7$ having tagger trustworthiness equal to T_{max}.

approximately T_{max} times more expensive than the max-flow computation of Advogato for a single trust level.

Another advantage of MaxTrust is that it considers the similarity between users, which can be viewed as pairwise trust that regulates the flow of total trust that the network assigns to a user. As a result, MaxTrust is able to incorporate additional information about the confidence each user can place on each other's tags, resulting in more accurate trustworthiness estimations.

MaxTrust computes the tagger trustworthiness $0 \leq w_i^t \leq T_{max}$ using a heuristic max-flow computation the cost of which increases linearly with T_{max}. In choosing T_{max} one has to consider the trade-off between computation cost and fine granularity in assigning trust values to users. MaxTrust proceeds in two phases: (a) the phase in which we transform the trust graph into a network flow problem; and (b) a heuristic suitable for approximating max-flow in case every user in the graph is directly connected to the sink.

Phase 1: Network Flow Graph Creation. In this phase, we transform the trust graph into an edge-capacitated network flow graph. We create an additional virtual supersource user (Figure 4). We then add an edge from the supersource to each trusted user $s \in S$. We add a directed edge from each user, except for the supersource, to an additional virtual supersink user. To prevent loops during the distribution of capacity among the users, we prune all edges that connect users at a higher distance from the supersource to users at a lower distance from the supersource. We also prune edges between users at the same distance from the supersource.

We now describe how we distribute capacity to the edges of the network flow graph. We denote as $C_{supersource}$ the sum of the capacity of the outgoing edges of the supersource. We set $C_{supersource} = (1 - p_d)|V| \cdot T_{max}$, where p_d is the portion of users in the trust graph $G(V, E_t)$ that are dishonest. We make the implicit assumption that we know the approximate number of honest users at the time we initialize the trust inference method. Next, we assign capacity $C_s = C_{supersource}/|S|$ to each edge from the supersource to each trusted user s. In the rest of this description, we denote as C_u the sum of the capacity of the incoming edges of user u.

Subsequently, we recursively assign capacities to the rest of the edges in the trust graph. That is, for each user u, we distribute $C_u - T_{max}$ capacity among the outgoing edges that connect u with its neighbors in the pruned graph. The capacity C_{uv} of the outgoing edge from user u to its neighbor v in the pruned graph is assigned proportionally to the tagging similarity ts_{uv} between user u and v: $C_{uv} = (C_u - T_{max})\frac{ts_{uv}}{\sum_{z \in F_u} ts_{uz}}$, where F_u is the set of u's friends. We also assign capacity T_{max} to the edge $u \rightarrow$ *supersink*. If $C_u < T_{max}$, we set $C_u = T_{max}$, and allocate no more capacity to u's neighbors. With this choice, we bias tagger trustworthiness towards higher scores for a smaller number of users, instead of lower scores for a larger number of users. This further limits the effectiveness of Sybil assertion poster attacks (Section 2.4).

If we overestimate the portion of dishonest users (p_d), then more honest users will be assigned low tagger trustworthiness resulting in the veracity of true assertions to decrease. If we underestimate it, then dishonest users will gain higher trustworthiness and they will be more potent in assigning higher veracity to fake assertions.

Phase 2: Max-Flow Computation. We now describe how we compute the maximum flow from the supersource to the supersink and derive the users' tagger trustworthiness. In our setting, edge capacity and flows take integer values. Thus, solving optimal max-flow with Edmonds-Karp (as done for Advogato) costs $O(T_{max}(1 - p_d)|V||E|)$, because it takes at most $C_{supersource} = T_{max}(1 - p_d)|V|$ augmentations. This is computationally prohibitive (Section 6.3), therefore we introduce a heuristic.

The heuristic executes T_{max} Breadth-First Search Operations (BFSO). The BFSO starts from the supersource. It *visits* every user i in the flow graph once in a BFS fashion. When the heuristic visits a user i, it *scans* i's children in a random order. For each child, it stores the last parent user that the BFSO visited before scanning the child. We denote the last visited parent of a scanned user j as $parent(j)$.

Thus the scanned nodes are the leafs of a tree whose root is the supersource. It scans the children in a random order in order to give all children the same probability to be accepted.

When the BFSO scans i's child j, it backtracks from j to the supersource through i as follows. First, it checks whether the edge $i \rightarrow j$ has at least capacity 1. If yes, it checks whether the capacity of the edge $parent(i) \rightarrow i$ is at least 1. If yes, it sets $i = parent(i)$ and repeats until $parent(i)$ is the supersource. If backtracking reaches the supersource, it adds 1 unit of flow to the edge $j \rightarrow supersink$. It also reduces the capacity of the edges along the backtracking path by 1. If the edges on the backtracking path upstream of i do not have at least capacity 1, the algorithm does not scan any more of i's children. This step costs $O(\Delta)$, where Δ is the graph diameter.

If the algorithm adds 1 unit of flow to the edge $j \rightarrow supersink$, j is considered for a subsequent visit, but is not considered for a subsequent scan by the same BFSO. If the algorithm does not add 1 unit of flow, j can be scanned from another parent. The BFSO continues until there are no more users to be visited.

After the BFSO ends, a new one starts from the supersource. It repeats this process until T_{max} BFSO are executed. The capacities and flows of the edges remain as adjusted during the previous BFSO. After T_{max} BFSO, the flow on the edge $j \rightarrow supersink$ corresponds to j's tagger trustworthiness.

The algorithm performs a total of $\Theta(T_{max}|E|)$ user scans. At each scan it performs $O(\Delta)$ capacity updates for each of the user's ancestors. Thus, our heuristic costs $O(T_{max}|E|\Delta)$. The diameter Δ of social graphs (small world networks) is typically

$O(\log(|V|)$ (measured to be 9 to 27 in real OSNs [Mislove et al. 2007]). In the evaluation of this article, our $200K$-user social graph has diameter 18.

Our heuristic takes advantage of the fact that all users are connected to the supersink. Thus, it finds in $O(1)$ an approximation of the shortest residual path to the supersink. It maintains the guarantees required by the trust inference method and offered by the optimal max-flow solution using Edmonds-Karp's algorithm: (a) if there is flow on a link $j \rightarrow \mathit{supersink}$, there will be flow on this link in the optimal solution; and (b) if there is flow on j's outgoing links there will be flow on the link $j \rightarrow \mathit{supersink}$. The heuristic misses the cases in which it would be preferable to not use ancestor capacity to accept a child j but to use it for another child m, because child j may have another parent that can pass flow to it, while child m does not. However, in our $200K$-user network flow graph this was not often the case, as indicated by the fact that the max-flow achieved with our heuristic was typically $\sim 96\%$ of the optimal max-flow. We analyze the assurances provided by the max-flow-based tagger trustworthiness mechanism in Section 4.2.

3.3.3. Periodic Update of Tagger Trustworthiness. MaxTrust computes the tagger trustworthiness of each node by using a tagging similarity-based trust graph. However, the social graph and the tags can change drastically over sufficiently long periods of time. To address the dynamicity of the tagging-similarity trust graph FaceTrust computes the tagger trustworthiness value by rerunning MaxTrust periodically, as often as it is computationally practical. Judging by the results in Section 6.3, we estimate that on a single off-the-shelf machine with 20–30GB memory, we could compute MaxTrust for a 10-million user social graph within 12 to 24h, thus allowing for daily recomputation. We use the periodically computed tagger trustworthiness values to compute the veracity scores on-demand, because the veracity computation (Eqs. (1), (2), and (3)) does not entail a significant overhead.

3.4. OSN-Issued Credentials

After the OSN provider (Section 2.3) is able to calculate the assertion veracity score for a user i's assertion A_i^t, the OSN provider can issue a relaxed credential for this assertion. As shown in Figure 1 and Figure 5, a credential issued by an OSN includes the assertion type t, the assertion A_i^t, and the assertion veracity score. The assertion veracity score is computed on-demand upon a verifier's request to view the credential.

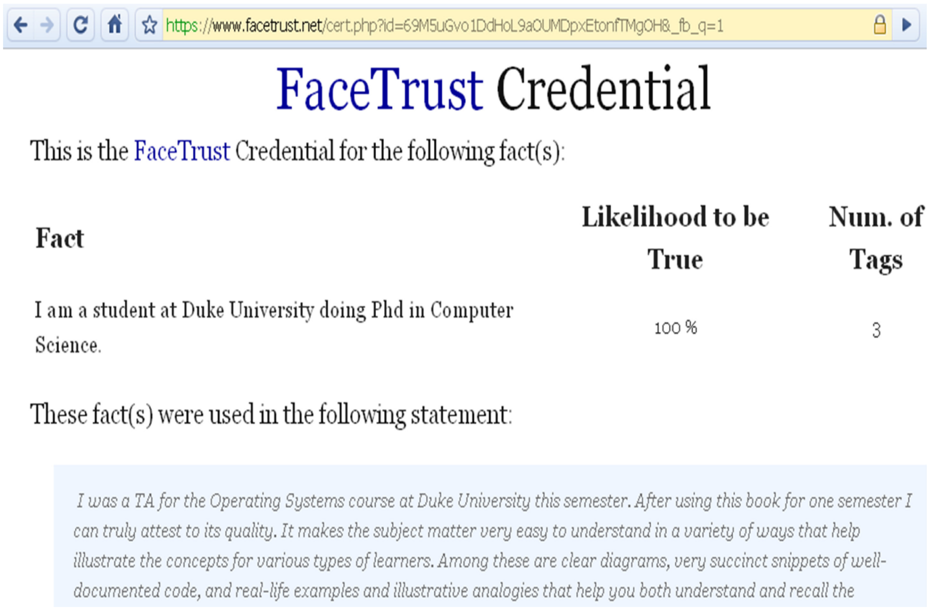
We use noncryptographic Web-based credentials that satisfy the goals listed in Section 2.5. Each credential as seen by the verifiers consists of:

— the `list of assertions` the user is certifying with their veracity scores and their types;
— content—an `excerpt` of the message (review, email, random string, etc.) for which the credential is used;
— context—a `URL` to or a `description` of the message for which the credential is used.

For example, a credential used for an online book review may include the following fields.

— [profession, CS professor, 100%, 17 tags]
— This is a great textbook and I highly recommend it ...
— http://www.amazon.com/Network-Design/product-reviews/100

This design binds a credential to the content and context it is used for, and ensures a credential's authenticity, as it cannot be used to verify the assertion in a different content and context. Using the Amazon review example, FaceTrust shows to the user


https://www.facetrust.net/cert.php?id=69M5uGvo1DdHoL9aOUMDpxEtonfTMgOH&_fb_q=1

FaceTrust Credential

This is the FaceTrust Credential for the following fact(s):

Fact	Likelihood to be True	Num. of Tags
I am a student at Duke University doing Phd in Computer Science.	100 %	3

These fact(s) were used in the following statement:

I was a TA for the Operating Systems course at Duke University this semester. After using this book for one semester I can truly attest to its quality. It makes the subject matter very easy to understand in a variety of ways that help illustrate the concepts for various types of learners. Among these are clear diagrams, very succinct snippets of well-documented code, and real-life examples and illustrative analogies that help you both understand and recall the

Fig. 5. Illustration of a human-readable FaceTrust credential for an online book review.

the URL of the credential (address bar in Figure 5). The user can subsequently add this URL to his review.

If a user wishes to obtain a credential to verify his identity attribute(s) when he posts a message on an online forum, he first requests a unique HTTPS URL or ID for the credential from the credential issuing Web site. He then copies the credential URL or ID into the message he wishes to use, publishes the message, and obtains the message URL (context). Subsequently, he selects the assertions he wishes to certify, and submits them together with an excerpt of his message (content) and the message URL (context). This completes the credential request process. The OSN provider creates a credential linked to the credential URL, which includes the selected assertions and their veracity scores, the excerpt of the message, and the message URL.

To ensure integrity, once a credential is created, the user cannot modify it. Our implementation also includes the number of total tags an assertion has in a credential to give a verifier more information for better judgment. If a credential has expired assertions, for every such assertion an expiration notice is included, along with the expiration date and the latest veracity score.

To verify the credentials of a user, a human verifier can follow the HTTPS credential URL and view the credential through her browser. The FaceTrust site employs a CA-signed certificate. The use of secure HTTP addresses most challenges the security of the Web channel. However, the system still remains vulnerable to phishing attacks if the verifier is not trained to expect an HTTPS page or does not properly verify the certificate.

In case the online message cannot include a credential URL, for example, Amazon strips URLs from user reviews, the user can instead include a specially formatted unique ID of the credential in his message when he requests the credential. A browser extension can read the specially formatted ID and prompt the user to view the credential.

In case a user wishes to use a credential to verify attributes to a human verifier with which he exchanges messages, the verifier and the user run the following protocol. The verifier sends to the user a randomly generated string. Subsequently, the user includes this string in the content field of the credential along with an excerpt of the message he wishes to send to the user.

FaceTrust's design also offers an XML Web service API, which can be accessed by online verifying services, such as Web sites that perform automated access control based on the user's identity attributes (e.g., age). The verifier presents a randomly generated string to the user, and the user generates a credential with the random string in the content field. Subsequently, the user posts to the online service the credential URL or ID that the user uses to certify his attributes, and the online service can retrieve the credential through an API call.

It is important to note that the credential does not reveal any personally identifiable information, unless the user has explicitly included such information in the assertion or the message. We also emphasize that the verification of the binding between the credential and the content and context it is issued for falls exclusively upon the verifying user or service. For example, the verifier is solely responsible for determining whether the content field in the credential matches the review or the random string.

In an earlier design of FaceTrust [Sirivianos et al. 2009], we proposed to use cryptographic anonymous, unlinkable, and nontransferable credentials [Camenisch and van Herreweghen 2002]. We discarded this design to obviate the need for user-side cryptographic operations, as it has been shown it is difficult for average users to use cryptography [Whitten and Tygar 1999].

However, with our solution, a user can ask a colluder that has a desired identity attribute to generate a credential for him; for example, an online verifying service would present the random string to the user, which the user would relay to the colluder who would create the credential and give its URL or ID to the user. Because the credentials are anonymous, the verifying service would have no way to verify that the claimed identity attribute belongs to the colluder and not the user. This collusion takes place without the colluder having to leave control of all his credentials to the user. On the other hand, with cryptographically nontransferable credentials, such collusion would require that the user becomes able to fully impersonate the colluder. The latter acts as a deterrent for the collusion between users in the case of cryptographic credentials. Although our solution does not entail such deterrent, we believe the fact that a new credential needs to be issued for each context (e.g., a new credential for every time a user enters an age-restricted site) makes such collusion sufficiently impractical.

3.5. Mitigating Sybil Assertion Posters

We now describe how we improve our scheme to defend against the Sybil assertion poster attack (Section 2.4). Under this attack, coalitions of dishonest users create a single or more Sybil accounts, post assertions on behalf of those Sybil accounts, and tag them as `true`. The dishonest users subsequently share the Sybil accounts and use their assertions to create FaceTrust credentials. Because honest users are not connected to the Sybil accounts and cannot tag their assertions, dishonest users do not need to tag differently from their honest friends. This results in high tagging similarity between honest and dishonest users. Subsequently, dishonest users do not have lower tagger trustworthiness than honest users and their tags on the false assertions are not discounted.

We simultaneously employ two techniques to mitigate this attack. The first technique addresses the case in which a coalition of dishonest users creates a single or a small number of Sybil accounts. We observe that colluders can create assertions on the few Sybil accounts, tag them as `true`, and use them unimpeded to present multiple false credentials. We mitigate this attack by imposing a quota on the number of credentials each account can issue. A reasonable approach in enforcing quotas is to impose an upper limit on the number of credentials a user can issue per month for each type of assertion, based on expected usage.

The second technique addresses the case in which the coalition of dishonest users creates multiple Sybil accounts to overcome the credential quotas. We modify the assertion veracity Eq. (1) as follows. Our solution relies on the assumption that honest users are typically both honest taggers and honest assertion posters. We can therefore use our Sybil-resilient tagger trustworthiness measure to infer how trustworthy their assertions are. To this end, we multiply the computed assertion veracity a_A of an assertion A posted by user j by a normalized value of the tagger trustworthiness of j.

$$a'_A = a_A \cdot min(1, c + (1-c)w^t_j/\overline{w}) \tag{3}$$

$\overline{w}$ is the tagger trustworthiness value for which $(1-p_d)|V|$ users have greater or equal tagger trustworthiness. p_d is the portion of users V in the trust graph $G(V, E_t)$ that are dishonest. c is a tunable parameter in [0, 1], that assigns a minimum veracity $a_A \cdot c$ to assertion A in case the tagger trustworthiness w^t_j of j is 0. The factor $min(1, c + (1-c)w^t_j/\overline{w})$ is 1 for taggers with trustworthiness higher than $\overline{w}$.

Because our trust inference method assigns very low tagger trustworthiness scores to multiple Sybils, this adjustment results in decreased veracity for assertions posted by colluders in this attack. Furthermore, because under several settings dishonest users have less tagger trustworthiness than honest users (Section 6.1.2), this equation results in further decreasing the veracity of false assertions.

3.6. Dealing with Underground Online Marketplaces for Tags and Credentials

FaceTrust may create perverse incentives that encourage the undesirable creation of a "Sybils as a Service" offering in underground markets. The main two offerings could be: (a) login credentials for Sybil accounts to enable their buyers to use the Sybils' credentials; and (b) Sybils that would tag assertions posted on the buyer's account to make them appear more truthful.

Offering (a) is essentially a manifestation of the Sybil assertion poster attack (Section 2.4). In Section 3.5, we describe how to address it. Additionally, in Section 6.1.3 we illustrate the effectiveness of our approach and its limitations when attackers deploy many Sybils to overcome the quota. We note, however, that in the underground marketplace setting, the buyer of the Sybil assertion poster accounts is likely to control an account that reflects his real persona and thus has sufficiently high tagger trustworthiness. The buyer is incentivized to connect his real account to the Sybil he purchased, assign high user-defined similarity to it, and tag similarly with it. The aim is for the Sybil to obtain higher tagger similarity yielding higher assertion veracity scores. Therefore, in this setting the Sybil assertion posters gain tagger trustworthiness not only from the dishonest accounts of their creators (as is the case for our evaluation in Section 6.1.3), but from the accounts of their purchasers too. Nevertheless, if the buyer acquires and connects to a small number of Sybils the credentials he can issue are limited by the quota. If, on the other hand, the buyer acquires and connects to a large number of Sybils (greater than 10, as can be seen in Figure 11(b)), MaxTrust ensures that the amount of tagger trustworthiness that flows to each Sybil is significantly diminished compared to when he connects to a few Sybils.

With regards to offering (b), FaceTrust does not have a in-built mechanism to defend against such activity. Sybil account buyers are actually incentivized to connect their honest user accounts to Sybils that promise to help them, which goes against our assumption that honest users do not befriend fakes easily. This allows Sybils to infiltrate the network and defeat MaxTrust. The only way to address this concern is to discourage users from befriending Sybils regardless of whether they perform desirable actions. We believe that such a discouragement is plausible because two users cannot be AIR friends unless they are also Facebook friends. This means that users who connect to Sybils expose themselves and their friends' Facebook information to unscrupulous entities. We believe this fact alone will dissuade many honest users from using offering (b). Furthermore, we note that such an underground service should clearly indicate that its Sybils perform this questionable activity, so that users know to connect to them. That is, this service would be announced on the Sybils' profiles and the underground online marketplace. This would make it easy for Facebook or the FaceTrust provider to use simple machine learning classifiers to detect and weed out such fraudulent AIR profiles.

4. ATTACK-RESISTANCE ANALYSIS

We now discuss the attack resistance of our design based on the assumptions, threats, and goals listed in Section 2.3, Section 2.4, and Section 2.5.

4.1. Assertion Credibility Analysis

First, we discuss the assurances that the assertion veracity Eq. (1) offers. For simplicity, we assume that the user j has higher than or equal tagger trustworthiness than $\overline{w}$, thus we do not need to consider the discounting introduced in Eq. (3).

THEOREM 4.1. *If the sum of the weights w_j^t of the dishonest users that have incorrectly tagged a false assertion by user i is less than* 0.75 *of the sum of the weight of the users that have tagged the assertion, the assertion will have less than* 0.5 *credibility.*

PROOF. We denote as R the ratio of the sum of the tagger trustworthiness of the dishonest users that have tagged i's false assertion over the sum of the tagger trustworthiness of the users that have tagged the assertion. The dishonest friends of i tag the false assertion as `true` (1). The honest users tag the assertion as `false` (−1). The goal of the system is to make the credibility of the false assertion be less than 0.5.
Using Eq. (1), we derive the inequality: $(R)1 + (1 - R)(-1) < 0.5 \Leftrightarrow R < 0.75$. □

The preceding inequality also means that honest users have to make sure that the sum of the tagger trustworthiness of their friends that will try to attack them by tagging their honest assertions as `false` does not exceed 25% of the total sum of their friends. This is a reasonable assumption for dishonest and honest users.

We now examine Eq. (2). The threshold M dictates how many dishonest users with a given tagger trustworthiness need to collude in order to make an assertion have nonzero veracity. A reasonable value for M is a multiple of the average tagger trustworthiness of honest users as derived by simulations (Section 6.1.2.) In the worst case, we need to consider dishonest users that have so far been tagging honestly and thus have obtained high tagger trustworthiness as honest users. Assuming that honest users have on average w tagger trustworthiness, at least M/w dishonest users have to tag an assertion in order for it to have nonzero veracity.

4.2. Tagger Trustworthiness Analysis

We now analyze the assurances offered by the trust inference method used to derive a user's tagger trustworthiness (Section 3.3.2). We observe that our max-flow-based trust inference method satisfies the *bottleneck property*: it ensures that the sum of the tagger trustworthiness of the dishonest user and its Sybils cannot exceed the sum of the capacity of the Sybil creator's incoming edges.

The purpose of our analysis is to derive the ratio R_{all} of the maximum sum of the tagger trustworthiness of dishonest users over the sum of the tagger trustworthiness of all users, when dishonest users are placed randomly in the social graph. Assuming that this ratio is on average maintained among the tagger trustworthiness of the friends of a user, R_{all} becomes a useful measure of the security of the assertion veracity, as is equivalent of the ratio R defined in Theorem 4.1.

We proceed by providing an upper bound M_d on the sum of the tagger trustworthiness of dishonest users and their Sybils. The tagger trustworthiness assigned to the Sybils of a dishonest user equals the sum of the flows on the edges connecting the Sybils to the supersink. It is upper bounded by the capacity C_u of the incoming edges of the dishonest user u that creates the Sybils. The closer the dishonest user is to the supersource, the higher the capacity of its incoming edges. This is an inherent issue for all trust metrics, including TrustRank [Gyöngyi et al. 2004] and Sumup [Tran et al. 2009]: the closer a node is to the trust seed users, the more effective its Sybil attack. The capacity of the incoming edges of the dishonest creator of Sybils depends on the distance from the supersource, the number of outcoming edges of users (fanout), and the capacity of the edges connecting the supersource to the trust seeds. Given this observation, the Sybil-limiting assurance of our method relies on how capacity is distributed among the graph edges.

For simplification, the following analysis assumes that the similarity-based trust graph consists of trees rooted at the trust seeds. We also do not consider tagging similarity in the edge capacity allocation during the transformation of the trust graph $G(V, E_t)$ to the network flow graph, thus assuming that all edges in E_t denote equal similarity. By not considering tagging similarity, we provide a pessimistic upper bound for M_d as the use of tagging similarity in assigning capacities results in the C_u of dishonest users to decrease. We also assume that there are no outgoing edges from dishonest users to honest users, which maximizes the tagger trustworthiness the Sybils of dishonest users can obtain.

As described in Section 3.3.2, the out-degree of the supersource is equal to the number of trust seeds $|S|$. We assume that the out-degree of all non-leaf users u is f. Each trust seed is the root of a subtree of $|V|/|S|$ nodes. V includes honest and dishonest users and does not include Sybils that dishonest users may create. Each user u is connected with an edge of capacity T_{max} with the supersink. Let p_d denote the portion of users in the similarity graph $G(V, E_t)$ that are dishonest. According to Section 3.3.2 (Phase 1), $C_{supersource} = T_{max}(1 - p_d)|V|$ and the capacity C_s of the edge from the supersource to a trust seed s is $C_{supersource}/|S|$. Also, $d = \lfloor \log_f C_s \rfloor$ denotes the maximum distance of users from a trust seed.

THEOREM 4.3. *For a given size $|V|$ of the trust graph, given out-degree f and given number of trust levels T_{max}, the maximum sum M_d of the tagger trustworthiness of dishonest users and their Sybils is only dependent on the portion of users that are dishonest p_d and the number of trust seeds $|S|$. It does not depend on the number of Sybils that dishonest users employ. M_d is expressed as follows.*

$$\begin{aligned} M_d = \; & p_d \cdot |S|(C_s \frac{(1-(1-p_d)^d}{p_d} + \\ & \frac{f\, T_{max}((f(1-p_d))^d - 1)}{(1-f)(f(1-p_d)-1)} + \frac{T_{max}((1-p_d)^d - 1)}{(1-f)p_d)}) \end{aligned} \quad (4)$$

PROOF. If a user u is at distance $1 \leq k \leq b$ from a trust seed s, the capacity of its incoming edges $C_u(k)$ is

$$C_u(k) = \frac{C_s - T_{max} \sum_{i=0}^{k-1} f^i}{f^k} = \frac{C_s - T_{max} \frac{f^k - 1}{f-1}}{f^k}. \quad (5)$$

We aim at determining the sum of the flow M_d that can be allocated to the links that connect dishonest users and their Sybils with the supersink when we solve the max-flow from the supersource to the supersink. M_d corresponds to the maximum sum of the tagger trustworthiness of the dishonest users and their Sybils. It is equivalent to the sum of the capacity of the incoming edges of the dishonest users. This is because in the tree topology we are considering, a dishonest user can ensure that all its incoming edge capacity is utilized by creating enough Sybils and connecting its outgoing edges to them.

We consider the case when dishonest users are placed randomly in the graph. This is in fact a pessimistic assumption that results in a higher M_d, because dishonest users are less likely than honest users to be placed close to trust seeds.

To derive the total maximum sum of the flow M_d, we sum up the capacities $C_u(k)$ of dishonest users across varying distances from the trust seed. Under our assumptions, the capacity of the incoming edge $C_u(k)$ of a dishonest user is assigned exclusively to Sybils or other dishonest users. Thus, when we account for the capacity of dishonest users at distance k, we should not account for the capacity of dishonest users that have dishonest ancestors.

At distance $1 \leq k \leq d$ from the trust seed, the average number of users (honest and dishonest) with no dishonest ancestor is $f^k(1 - p_d)^{k-1}$. Correspondingly, the number of dishonest users at distance k with no dishonest ancestor is on average $p_d f^k (1 - p_d)^{k-1}$. We derive that the maximum sum of the tagger trustworthiness M_d of dishonest users when dishonest users are placed randomly in the social graph and utilize all their capacity C_u to provide trust flow to dishonest users and Sybils is

$$M_d = p_d \cdot |S| \sum_{k=1}^{d} f^k (1 - p_d)^{k-1} C_u(k). \quad (6)$$

The previous equation is equivalent to Eq. (4). □

Let $D \subset V$ denote the set of dishonest users randomly distributed in the social graph. The average sum of the tagger trustworthiness of users in D is

$$(\lfloor \log_f |V| \rfloor - 2) \frac{|D| C_{supersource}}{|V|}. \quad (7)$$

The honest users in $V \setminus D$ obtain the remaining capacity.

$$C_{supersource} - (\lfloor log_f |V| \rfloor - 2) \frac{|D| C_{supersource}}{|V|} = C_{supersource}(1 - (\lfloor log_f |V| \rfloor - 2)|D|/|V| \quad (8)$$

We can now provide the ratio R_{all} of the maximum sum of the tagger trustworthiness of dishonest users over the sum of the tagger trustworthiness of all users, when dishonest users are placed randomly in the social graph and utilize all their capacity to provide flow to dishonest users and Sybils.

$$R_{all} = \frac{M_d}{C_{supersource}} = \frac{M_d}{T_{max}|V|(1 - p_d)} \quad (9)$$

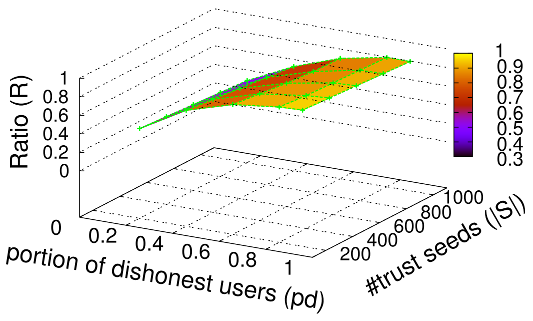


Fig. 6. Ratio R_{all} of the maximum sum of the tagger trustworthiness of honest users over the sum of the tagger trustworthiness of all users, when dishonest users utilize all their capacity to provide flow (tagger trustworthiness) to dishonest users and Sybils. $|V| = 200K$, $T_{max} = 10$ and $f = 10$.

According to Eq. (1), a good ratio R_{all} to prevent dishonest users from making their assertions to obtain a veracity score higher than 0.5 is on average 0.75. As can be deduced from the preceding equations and is illustrated in Figure 6, the ratio R_{all} increases substantially as a function of the portion of dishonest users p_d and decreases slightly as a function of the number of trust seeds $|S|$. It does not depend on the number of Sybils that dishonest users employ. Under the examined setting ($|V| = 200K, f = 10$, $T_{max} = 10$), increasing $|S|$ from 100 to 1000 results in R_{all} decreasing by ~ 0.1. In addition R remains below 0.75 as long as p_d is below 0.4.

The ratio R is large even when p_d is small (e.g., $R = 0.44$ when $p_d = 0.2$ and $|S| = 400$) because our model examines the worst-case scenario: a dishonest user never distributes capacity to children that are honest. This corresponds to dishonest users assigning 0 tagging similarity to their honest friends, while their honest friends assign 1.0 tagging similarity to them. Under our model, as the number of trust seeds $|S|$ increases, honest users have more opportunities to receive capacity from the trust seeds or other honest users. This is the reason we observe the decrease of R_{all} as $|S|$ increases.

Our simulation-based evaluation (Section 6.1.2) considers a real social graph (under which honest users can be connected to trust seeds via multiple paths), thus it yields higher tagger trustworthiness for honest users.

Under our tree topology model, the existence of multiple seeds also mitigates the impact of a focused adversary that manages to be close to a trust seed and establish high tagging similarity with it. This is because this focused dishonest user at distance k from the trust seed obtains at most $(C_{supersource}/|S| - T_{max}\frac{f^k-1}{f-1})/f^k$ capacity for its incoming edges instead of $(C_{supersource} - T_{max}\frac{f^k-1}{f-1})/f^k$.

5. IMPLEMENTATION

We implemented FaceTrust as a three-tier Web application so that we can evaluate the design and its assumptions using a real-world deployment. The front-end of FaceTrust is the "Am I Really?" Facebook application, which is implemented using the PHP Facebook developer API. The FaceTrust application server serves the HTTP content, collects assertions and tags, as well as maintains the social graph. The FaceTrust server employs MySQL to store the user, assertion, tagging, veracity, and social graph information. It uses a Java implementation of MaxTrust.

Game with a purpose. We built the "Am I Really?" (AIR) Facebook application using a "game with a purpose" design to incentivize social tagging. AIR is a "micropolling" application. Users post facts about themselves and ask their AIR friends, which are also their Facebook friends, to tag them as `true` or `false`. For example, a user may post the question "Am I really older than 18?" or questions of lighter nature, such as "Am I really good at baseball?" AIR users can view the veracity of a friend's assertion only after they have tagged his assertion, and after the assertion amass a threshold number of tags (3 in our implementation).

According to Facebook's terms of use, we are not allowed to store long term the friends of a user that the Facebook API provides. To circumvent this restriction, AIR asks a user to declare which of his friends he would like to have as friends on AIR. To further ensure that connections in the AIR social graph correspond to real-life acquaintances (Section 2.3) and to address the problem of promiscuous users that naively establish Facebook connections with malicious users, we explicitly ask a user to declare as AIR friends only persons with whom he has met in person.

We also built a credential issuing Web site `https://www.facetrust.net` [FaceTrust-Credentials 2014], which links each user's account with her Facebook account. A user that wishes to prove an identity attribute to other users or online services requests a credential as we describe in Section 3.4. The issuing Web site pulls the user's assertions and her veracity scores from the AIR database back-end.

6. EVALUATION

We evaluate the following aspects of FaceTrust.

Effectiveness. How strongly do assertion veracity and tagger trustworthiness correlate with the truth, and how well does the design withstand incorrect user tagging and colluder and Sybil attacks?

Practicality and usage. How often and how accurately does a user tag his friends to help them obtain credentials?

Computational feasibility. A social network may consist of several hundreds of millions of users. Will an OSN provider have sufficient computational resources to mine the social graph and derive tagger trustworthiness scores?

We use simulations on a sample Facebook social graph and a real-world deployment to answer these questions. We discuss each aspect in turn.

6.1. Effectiveness

We first examine whether true assertions obtain high veracity and false assertions obtain low veracity, even in the presence of dishonest users and Sybil attacks. We also study the limits of our approach, that is, under which conditions and attack strategies false assertions can obtain high veracity.

We start by evaluating the ability of our max-flow-based trust inference scheme, MaxTrust (Section 3.3.2), to assign low trustworthiness to dishonest users and Sybils. We then proceed to analyze the effectiveness of the assertion veracity mechanism (Section 3.2), which weighs user tags based on tagger trustworthiness. In this evaluation, we also compare the effectiveness of MaxTrust to another max-flow-based trust inference, Advogato.

For a more realistic evaluation, we use a crawled sample of the Facebook social graph, which consists of a 200K-user connected component obtained from a 1M-user sample [Gjoka et al. 2010] via the "forest fire" sampling method [Leskovec and Faloutsos 2006]. The average and maximum number of friends of each user in the graph is $\sim$ 24 and 313, respectively. The diameter of this graph is 18 and the clustering coefficient is 0.159.

6.1.1. General Simulation Settings. Each user in the social graph posts a single assertion of the same type on his profile. Honest users always post true assertions and dishonest users always post false assertions. Furthermore, the honest users tag as `true` the assertions posted by their honest friends and as `false` the assertions posted by their dishonest friends. The dishonest users tag all assertions as `true`, regardless of whether they are true or not. By doing so, dishonest users collude to increase the veracity of each other's assertions. When dishonest users behave in exactly the opposite way honest users do, they become disconnected from the honest nodes in the tagging-similarity-based flow graph. By truthfully tagging the assertions of honest users, dishonest users attempt to have common tags with other honest users in order to increase their tagging similarity with trustworthy users. This is a manifestation of the tagger camouflage attack (Section 2.4). Both honest and dishonest users are randomly distributed in the social graph, unless specified otherwise. The case of dishonest colluders that form coalitions in the social graph is discussed in Section 6.1.3. In addition, each user tags the assertions of at most F of his friends. We vary F to reflect various degrees of adoptability of social tagging.

To obtain the tagger trustworthiness, we use MaxTrust described in Section 3.3.2 and Advogato. Throughout the result description, MT and AD means MaxTrust and Advogato, respectively. We obtain the tagger trustworthiness as described in Section 3.3.2. We do not consider the user-defined similarity (Section 3.3.1), as we model no notion of a priori trust between users. We set $T_{max} = 100$ (Section 3.3.2). For each experiment, the minimum sum of the trustworthiness of taggers M (Section 3.2) is equal to the average tagger trustworthiness of honest users. When 50% of the users are honest, $F = 20$ and we use 1000 randomly selected trust seeds; this is 47.5. We set $c = 0,2$ (Eq.(3)). We vary the sum of the capacity of the outgoing edges of the supersource, $C_{supersource}$, depending on the portion of dishonest users. That is, $C_{supersource}$ increases as the portion of dishonest users decreases (Section 3.3.2). For MaxTrust, we employ 1000 trust seeds, which are randomly selected among the honest users. For Advogato, we randomly select a single trust seed. We repeat each experiment 5 times (with a different set of honest/dishonest users, tags, and seeds) and plot the mean and the 95% confidence intervals (too small and not visible in most configurations).

6.1.2. Tagger Trustworthiness Effectiveness. As described in Section 3.2, the tags on assertions are weighted by their tagger's trustworthiness. Therefore, we first need to examine the effectiveness of tagger trustworthiness (Section 3.3) under various strategies employed by dishonest users. We consider the tagger trustworthiness scheme effective if: (a) it assigns substantially lower trustworthiness to Sybil users than to honest users; and (b) it does not assign higher trustworthiness to dishonest users than to honest ones.

Dishonest users do not employ Sybils. In this series of experiments, dishonest users do not employ Sybils. In Figure 7(a), we observe that the trustworthiness of honest users (MT-Honest) is substantially higher than the one of dishonest users (MT-Dishonest) when the portion of honest users is small. The reason is that honest and dishonest users differ in terms of tagging. When the portion of honest users is relatively low and honest and dishonest users are placed randomly, there are many opportunities for honest and dishonest users to tag dissimilarly. Because tagging similarity captures the difference in tagging behavior between dishonest and honest users, this translates to low pairwise trust between them. In addition, because trust is seeded at honest users, MaxTrust's transitive trust mechanism assigns lower tagger trustworthiness to dishonest users.

When the portion of dishonest users is small, for example, 10%, dishonest users have almost equal trustworthiness to the honest ones. But the eventual measure of

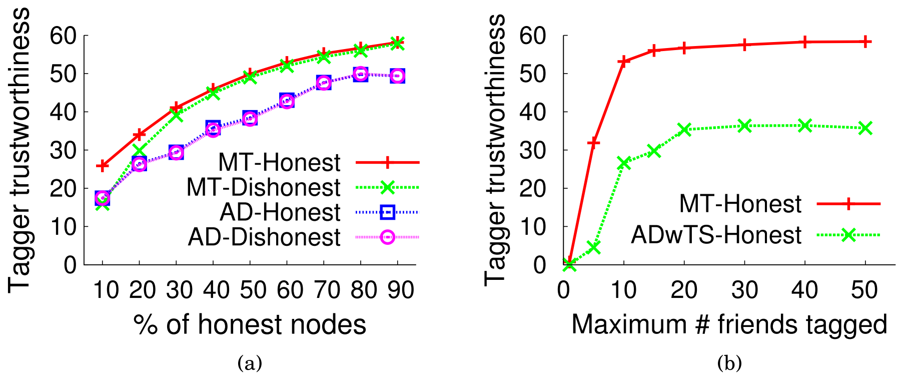


Fig. 7. Mean tagger trustworthiness in range [0,T_{max} = 100]: (a) as a function of the fraction of honest nodes when the maximum number F of friends a user tags is 20; (b) as a function of F when 80% of users are honest; dishonest users do not employ Sybils. MT = MaxTrust, AD = Advogato, ADwTS = Advogato with tagging-similarity-based trust graph.

interest is the assertion veracity of fake assertions, and this score is low, as desirable. This is because there are more tags from honest users than there are from dishonest users (Figure 9(a)). At the same time, we observe that when the honest users are less than 30% (which means assertions are likely to have more fake tags), they obtain substantially higher trustworthiness than dishonest ones. This results in the veracity of false assertions being significantly lower than the veracity of true ones (Figure 9(a)), even under this extreme attack scenario.

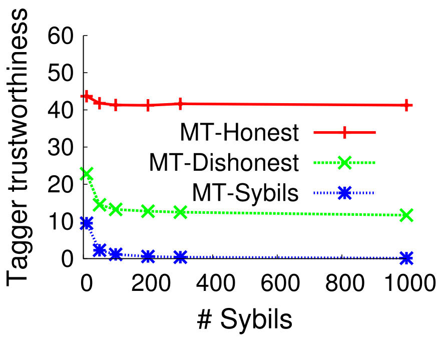
On the other hand, Advogato assigns almost equal trustworthiness to both honest (AD-Honest) and dishonest users (AD-Dishonest), regardless of the portion of dishonest users. It is because Advogato does not consider the tagging similarity while conducting the max-flow computation. This result demonstrates the importance of tagging similarity, which results in dishonest users having less influence on the system's operation.

In Figure 7(a), we observe that the average tagger trustworthiness increases as the portion of dishonest users decreases. Because $C_{supersource}$ increases as the number of dishonest users decreases, more users can obtain higher value of tagger trustworthiness.

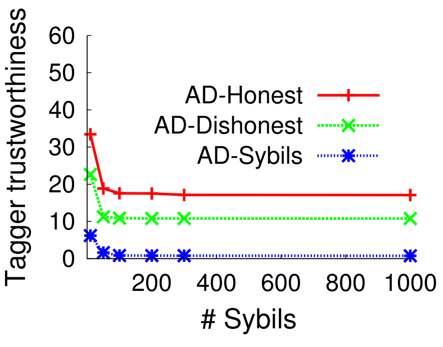
Figure 7(b) shows the trustworthiness of honest taggers as a function of the maximum number of friends F each user tags. This figure illustrates the importance of F. As F increases, the number of common tags C_t (Section 3.3.1) used to derive the tagging similarity increases. For $F < 10$, the tagging similarities between users are almost 0 and the similarity-based trust graph is disconnected, resulting in honest users getting very low trustworthiness. As F increases, the trust graph becomes more connected and honest users obtain increased tagger trustworthiness.

When tagging is infrequent, a large fraction of edges between honest users do not have high tagging similarity, as it becomes less likely for honest users to tag the same assertions. As a result, honest users get relatively low tagger trustworthiness. As can been seen in Figure 7(b), for honest users to achieve high tagger trustworthiness, F should be ≥ 10.

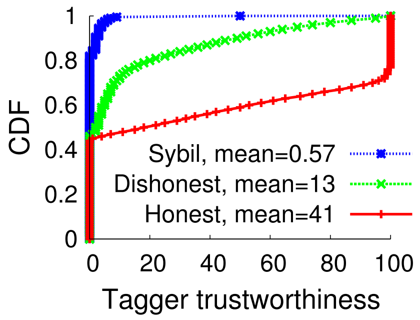
In Figure 7(b), we also observe that Advogato yields a lower mean of tagger trustworthiness for honest users than MaxTrust, even if we incorporate tagging similarity in allocating the capacities of Advogato's network flow graph. The main reason is that Advogato uses a single trust seed and the coverage of the trust propagation depends on the position of the seed in the graph.



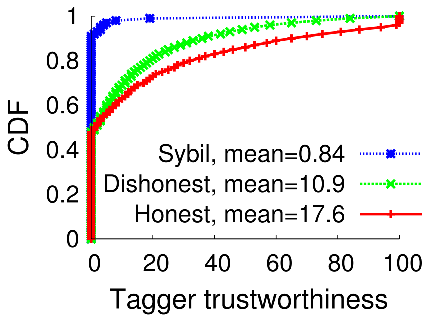
(a) MaxTrust



(b) Advogato



(c) MaxTrust



(d) Advogato

Fig. 8. (a,b) Mean tagger trustworthiness as a function of the number of Sybils each dishonest user creates; (c,d) CDF(Cumulative Distribution Function) of the tagger trustworthiness of honest, dishonest, and Sybil users; dishonest users employ 200 Sybils each, $F = 20$ and 50% of users are honest.

Dishonest users employ Sybil Taggers. To evaluate MaxTrust's resilience to Sybil attacks, all the dishonest users create a varying number of Sybils. All the Sybils are connected to their creator and are fully connected to each other. Sybils tag the false assertions of their creator as `true` to increase the veracity of those assertions. The creator always has tagging similarity 1.0 with all its Sybils. This corresponds to the configuration that maximizes the tagger trustworthiness of Sybils.

As can be seen in Figure 8(a), when the number of Sybils is 200, the tagger trustworthiness of Sybils is on average 72 and 22 times lower than the trustworthiness of honest and dishonest users, respectively. This is due to the bottleneck property of our trust inference mechanism (Section 3.3.2), which limits the amount of trust that can be assigned to the Sybils of a dishonest user. While dishonest users that do not employ Sybils get a similar trustworthiness score with honest users (Figure 7(a)), the trustworthiness of dishonest users employing Sybils is on average 3 times lower than the trustworthiness of honest users.

We also observe that as the number of Sybils increases from 10 to 200, the tagger trustworthiness of honest and dishonest users also decreases by 12% and 30%, respectively. The reason is that the incoming flow that passes through a dishonest user is now assigned to the Sybil users instead of other users downstream. Nevertheless this decrease is not substantial, especially for honest users, because there are multiple trust paths through which flow can reach these users.

In Figure 8(b), we observe that Advogato can also assign lower tagger trustworthiness to dishonest users employing Sybils than to honest users. However, the tagger

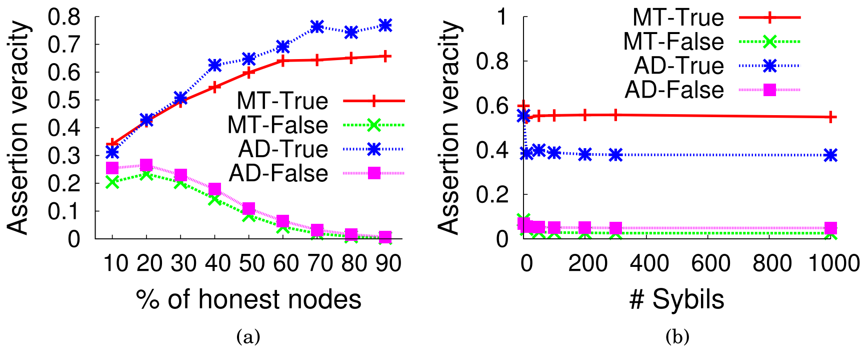


Fig. 9. Mean veracity of true and false assertions when $F = 20$: (a) as a function of the fraction of honest nodes when dishonest users do not employ Sybils; (b) as a function of the number of Sybils per dishonest user when 50% of users are honest.

trustworthiness of honest users decreases substantially as the number of Sybils increases. The reason is that Advogato does not use the tagging similarity trust graph, therefore allowing for more trust to flow to dishonest users and their Sybils. In this case, the numerous Sybils have their trustworthiness increase by a small margin, whereas the trustworthiness of honest users decreases drastically.

We now examine how tagger trustworthiness is distributed among the users. Figure 8(c) and Figure 8(d) depict the CDF(Cumulative Distribution Function) of the tagger trustworthiness of honest, dishonest, and Sybil users of MaxTrust and Advogato, respectively. As can be seen, for both of MaxTrust and Advogato, almost 80% of Sybil users have 0 tagger trustworthiness. However, while there is substantial variance in the trustworthiness scores of honest and dishonest taggers for MaxTrust, the trustworthiness distributions of honest and dishonest taggers are similar to each other for Advogato.

Because we set the assertion veracity threshold (Eq. (2)) to be close to the mean tagger trustworthiness, the tagger trustworthiness distribution greatly affects how the assertion veracity is computed. In MaxTrust we can safely set the threshold equal to the mean tagger trustworthiness for that portion of honest users, and in many cases the honest users that tag an honest assertion have sufficient tagger trustworthiness.

Assertion poster camouflage attack. We also evaluate the resilience of FaceTrust when dishonest users use the assertion poster camouflage attack (Section 2.4). Honest users post one true assertion, and dishonest users post one false assertion with a varying number (1–10) of true assertions for camouflage. We obtain the average tagger trustworthiness of honest and dishonest users under 80% honest nodes, 200 Sybils per dishonest users, and $F = 20$. We observe that the effect of the camouflage attack on the resulting tagger trustworthiness is insignificant. For the sake of conciseness, we opted not to plot this result.

Although it is not demonstrated in our experimental setting, this attack can be further mitigated by assigning distinct tagger trustworthiness scores for each type. As a result the camouflage attack cannot occur for many assertions of the same type because the tagging similarity of the user's dishonest friends with honest friends for that particular type will drop, and so will his tagger trustworthiness.

Tagger trustworthiness evaluation conclusions. The aforesaid results illustrate that under our tagging-similarity-based trust inference mechanism (MaxTrust) dishonest users obtain substantially lower trustworthiness than honest users when the portion of honest users is small. In addition, we show that Sybil users obtain almost two

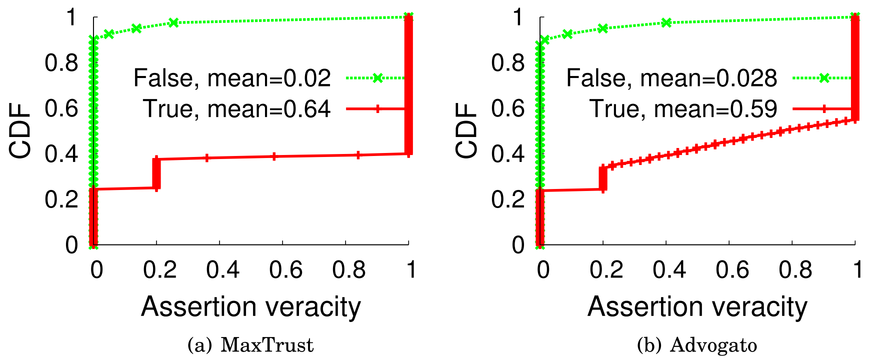


Fig. 10. CDF (Cumulative Distribution Function) of assertion veracity when dishonest users employ Sybils.

orders of magnitude less trustworthiness, under common Sybil strategies. Our results have also illustrated the importance of the frequency of tagging, as modeled by the parameter F.

6.1.3. Assertion Veracity Effectiveness. The assertion veracity scoring is dependent on the mechanism for determining the weight of the taggers, which we evaluated in the previous section. We now evaluate the assertion veracity computation technique itself (Section 3.2) under varying attack scenarios.

Dishonest users do not employ Sybils. Figure 9(a) plots the mean veracity of honest and false assertions as a function of the portion of honest users, when dishonest users do not employ Sybils. We observe that when the fraction of honest users exceeds 50%, the mean veracity of true assertions substantially exceeds that of false ones. Unlike plain majority voting, MaxTrust assigns low veracity to false assertions even when the fraction of dishonest users is large. We computed the Pearson correlation coefficient between the veracity values for MaxTrust in Figure 9(a) and the ground truth (true or false) and we found a strong correlation of 0.91. We also observe that Advogato achieves similar performance to MaxTrust. As the portion of dishonest users increases, the number of users that obtain very low tagger trustworthiness increases. Consequently, multiplying the veracity of an assertion by the normalized tagger trustworthiness of its poster (Section 3.5) decreases the veracity score of both true and false assertions.

Dishonest users employ Sybil taggers. Figure 9(b) shows the veracity of true and false assertions when dishonest users employ Sybils. Each dishonest user creates a varying number of Sybils. The Sybils are connected only to their creator and tag all its assertions as `true`. As can be seen, the dishonest users gain little benefit by using Sybils in our setting. Although there are many Sybil taggers for false assertions, most of them have very low (or 0) tagger trustworthiness and the sum of tagger trustworthiness of Sybil taggers is most often below the threshold M (Eq. (2)).

Figure 9(b) also shows that the veracity of true assertions is affected by Sybils to a larger degree when we use Advogato than when we use MaxTrust. Although Advogato can successfully suppress the tagger trustworthiness of Sybil taggers, it also decreases the tagger trustworthiness of honest users. Consequently, some true assertions do not have enough taggers to pass the threshold M (Eq. (2)) and they cannot obtain sufficiently high assertion veracity.

Figure 10(a) shows how veracity is distributed among true and false assertions when we use MaxTrust. We depict the CDF of the assertion veracity of all $200K$ assertions. We observe that 60.6% and 14.3% of true assertions obtain veracity equal to 1 and

0.2, respectively. Also, 24.3% of true assertions obtain 0 veracity. The true assertions with $c = 0.2$ veracity belong to honest users with 0 tagger trustworthiness (Eq. (3)). The true assertions with 0 veracity are the ones for which the sum of their taggers' veracity scores are below M. The number of these incorrectly assessed true assertions can be reduced by increasing the maximum number of friends that users tag (F), that is, increasing the adoption of social tagging. Incorrectly assessed assertions can be further avoided by designating more trust seeds. Unlike true assertions, most of the false assertions, 90%, obtain 0 veracity. Only 1.5% of false assertions obtain veracity 1. This result suggests that FaceTrust's assertion veracity scoring mechanism is effective, but not absolutely accurate. Thus, it should not be used to control access to critical resources.

To compare MaxTrust to Advogato, Figure 10(b) shows how veracity is distributed among true and false assertions when we use Advogato. In Advogato, we observe that only 43% of true assertions obtain veracity equal to 1. That is, Advogato can prevent false assertions from obtaining high veracity, but it also suppresses the veracity of true assertions.

Dishonest focused colluders. We also evaluate the case in which dishonest users form a coalition. The dishonest colluders in a group are connected to each other and tag each other's assertions as `true`. This experiment differs in that it is guaranteed that each dishonest user has a specified minimum number of dishonest colluders. This corresponds to a more focused and coordinated attack. Figure 11(a) depicts the mean veracity of the assertions posted by the dishonest users as a function of the size of a coalition.

We observe that the false assertions of colluders can get higher average veracity than the true assertions only if the coalition size exceeds a relatively high threshold (30 for MaxTrust). This is due to: (a) the increased number of dishonest taggers; and (b) the increased tagger trustworthiness of the colluders. The tagger trustworthiness of colluders increases because users closer to seeds can get higher tagger trustworthiness in MaxTrust. If a single colluder in a group is close to a trust seed, all the colluders in his group, which are connected to him, may get high tagger trustworthiness. As the number of colluders increases, both sources of increased tag weight become more prominent and the assertions of colluders get high veracity. We also observe that Advogato performs substantially worse than MaxTrust; the threshold over which false assertions acquire higher veracity than true ones is reduced to 15 instead of 30.

The preceding results reveal a limit of our approach. If a substantial number of colluders coordinates, they can ensure that their assertions have high veracity. Nevertheless, rational colluders need to expend effort (Section 2.4), which may discourage them from orchestrating an attack. Hence, FaceTrust credentials should not be treated as the absolute truth and they should instead be used as an additional indication of veracity.

Dishonest users employ Sybil assertion posters. We now evaluate our system when coalitions of dishonest users perform the Sybil assertion poster attack (Section 2.4). Each dishonest user creates Sybils to which all the colluders in its coalition connect. The Sybil users post assertions and all the colluders tag them as `true`. At the same time, dishonest users tag honestly for all other assertions in an attempt to establish high similarity with honest users. This attack is equivalent to dishonest users who choose to connect only to other dishonest colluders. As explained in Section 2.4, this attack results in the colluders having access to assertions that cannot be voted as `false` by other honest users, thus those assertions are expected to have high veracity.

In Figure 11(b), when the number of Sybil assertion posters is small, for example, 10 for MaxTrust, we observe that the veracity of false assertions is higher than that of

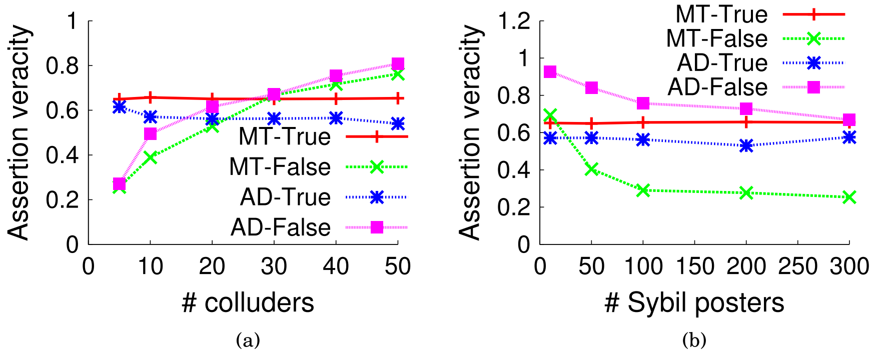


Fig. 11. (a) Mean veracity of assertions posted by dishonest colluders as a function of the coalition size; 80% of users are honest and $F = 20$; (b) mean veracity of false assertions posted by Sybil posters as a function of the number of Sybil assertion posters in the group, when 80% of users are honest, $F = 20$, and the group size is 30.

true ones. Because the number of Sybils is small, MaxTrust does not assign low tagger trustworthiness to them. Consequently, Eq. (3) (Section 3.5) does not mitigate this attack, because both the colluding dishonest taggers and the Sybil posters have relatively high tagger trustworthiness. This result reveals another limit of our approach. Nevertheless, FaceTrust prevents dishonest users from using the assertions of those Sybils in multiple contexts by imposing a quota (Section 3.5) on the number of credentials each user can request.

Figure 11(b) also shows that MaxTrust is significantly more resilient to the Sybil poster attack than Advogato. Unlike MaxTrust, in Advogato false assertions obtain higher veracity than true ones as long as the attacker employs less than 300 Sybil assertion posters. Because there is a single seed in Advogato, colluders located near the seed can pass high tagger trustworthiness to their Sybil users, thus boosting their false assertions.

When the colluders create many Sybils to overcome the quotas, they have to cope with the fact that the tagger trustworthiness of the Sybils is reduced. Consequently, the mean assertion veracity is reduced as shown in Figure 11(b). This result indicates the importance of multiplying the assertion veracity by the poster's tagger trustworthiness as described in Section 3.5. Furthermore, rational dishonest users incur a cost to create Sybils, such as solving CAPTCHAs during Facebook account registration in Section 2.4, which further limits their ability to subvert our scheme.

Assertion veracity evaluation conclusions. The previous evaluation illustrates that our assertion veracity scoring technique results in false assertions obtaining substantially lower veracity than true ones. We show that this holds even under commonly deployed attack strategies (Sybils and colluders). In addition, we demonstrate the limits of our approach by explicitly describing elaborate colluding attacks that FaceTrust does not sufficiently mitigate.

6.2. Facebook Deployment

FaceTrust requires a new form of user input: assertions and tags. In addition, in order for the veracity scores to correlate positively with the ground truth, it requires trustworthy users to tag honestly and similarly. These facts motivate us to ask: Are users willing to tag their friends' tags? How often and honestly will they tag? To answer these questions, we deployed the "Am I Really?" (AIR) Facebook application (Section 3.1) for users to post and tag assertions, and advertised it on Facebook. The Facebook advertisements resulted in approximately 100 installations of AIR.

We collected a dataset consisting of 1108 real Facebook users.[1] Specifically, 395 of those users chose to declare that they are friends with at least one AIR user, thus having one or more neighbors in the AIR social graph. For the rest of this evaluation we provide statistics concerning those 395 users, because they are the only ones that can tag friends in AIR. Our dataset includes 2410 social connections established between September 1st, 2009 and January 10, 2010. These connections form several connected components, the largest of which includes 182 users. The average number of friends a user has in that largest component is 3.8 and the diameter of the component is 4. Our live system computes tagger trustworthiness scores using MaxTrust. We employ 10 trust seeds, set $T_{max} = 10$, and assume that 90% of the network consists of honest users. We incorporate user-defined similarity (Section 3.3.1) in the computation of tagging similarity, using $b = 5$. We again set $c = 0.2$ (Section 3.5). On average, friends have N_t (as defined in Section 3.3.1) equal to 5.2, 10.4, 3.7. and 2.8 for type t of age, location, profession, and gender, respectively.

To protect user privacy, we anonymize all Facebook and AIR-specific identifiers and exclude the assertions that include personally identifiable information prior to data processing. In addition, the application informs its users that their personal data will not be published.

Figure 12(a) shows the complementary cumulative distribution (CCDF) of users as a function of the number of tags they post. We observe that even in this small social graph, more than half of the users have tagged at least 8, 6, 4, and 1 time for assertions of type age, profession, location, and gender, respectively. We also find that users tag on average 14.4, 10.4, 7.5, and 4.6 times for assertions of type age, profession, location, and gender, respectively. We believe that when the system is widely adopted, users will have on average many more friends to tag. Thus, we speculate that the number of assertions users tag is likely to exceed 10, which is the number needed to obtain accurate tagger trustworthiness (Figure 7(b) in Section 6.1.2).

Figure 12(b) shows the CCDF for the number of assertions users post for each assertion type. More than one quarter of the 395 users have posted at least 8, 6, 4, and 2 assertions of type age, profession, location, and gender, respectively. We also find that users post on average 5.6, 3.6, 2.6, and 0.9 assertions of types age, profession, location, and gender. This is indicative of the fact that users use this application as intended and do not feel uncomfortable reporting such information to their friends and FaceTrust. We also observe that users tend to post more assertions that concern their age or profession than their location. Users are probably less motivated to ask others for their opinion on their location.

We now examine the AIR profiles of 10 users that were randomly selected out of the 395 users. For these 10 users we used out-of-band means (in-person questions and close examination of their profiles) to determine the ground truth for their age, gender, location, and profession assertions. We collect a total of 50, 50, 50, and 20 age, profession, location, and gender assertions, respectively. These include 14 false age assertions, 21 false profession assertions, 19 false location assertions, and 10 false gender assertions. Each of these assertions were tagged ~ 6 times on average by distinct users.

Figure 12(c) shows the mean veracity per type of the true and false assertions with and without attackers in the system. Per each type, the first column depicts the mean assertion veracity of true assertions. The second column depicts the mean veracity of false assertions in the absence of attackers. The third column shows the mean veracity

[1]Duke University IRB Protocol 3015.

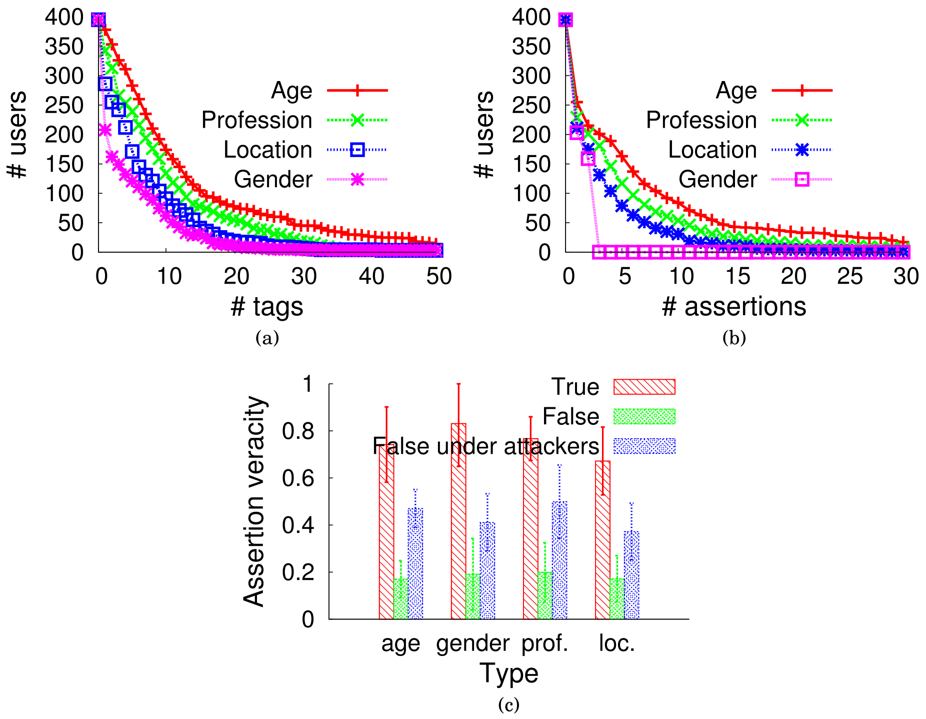


Fig. 12. (a) CCDF of the number of "Am I Really?" users as a function of the number of tags per user for each assertion type; (b) CCDF of the number of users as a function of the number of posted assertions per user for each assertion type; (c) veracity per type of true and false assertions in FaceTrust's real-world deployment with and without attackers; the error bars denote 95% confidence intervals.

of false assertions when we inject 20 dishonest users in AIR's social graph. The aim is to emulate an attack during which the injected dishonest users collude with the 10 randomly selected users to make their false assertions appear `true`. The injected dishonest users do not represent real Facebook accounts. They are connected to the 10 randomly selected honest users, such that each of these real users is AIR friends with two distinct dishonest users. That is, we start by connecting the first honest user with two randomly selected dishonest users. We then remove those two dishonest users from the pool of users to connect to and repeat the process with the next honest user. The dishonest users tag the false assertions of the 10 real users as `true`. In an attempt to increase its similarity with honest users, a dishonest user launches the camouflage attack by tagging all the other assertions as `true`, if their prior veracity of the assertion was greater than 0.5 and `false` otherwise.

In Figure 12(c), we see that the computed veracity for true and false assertions in the absence of attackers correlates very well with the ground truth. This result indicates that users tend to tag correctly. We observe that users may make some mistakes in assessing each other's age, but when the truth for an assertion is straightforward, such as for gender, the veracity of the assertion is high.

As can be seen in the third column for each type, the injected dishonest users have boosted the veracity of false assertions. This is mainly because the AIR social graph is small, with each honest user having less than 4 honest friends on average. Two attackers per user caused the false assertions' veracity to increase substantially. However, there is still a distinguishable gap between the average veracity of true and false assertions, indicating the resilience of FaceTrust's assertion veracity scoring mechanism.

6.2.1. Deployment Conclusions. Our Facebook deployment results indicate that users tag sufficiently frequently for the tagger trustworthiness measure to be effective. Importantly, our results also indicate that benign users tag mostly correctly. This demonstrates the efficacy of relying on users to certify each other's identity attributes.

6.3. Computational Efficiency

We have benchmarked the computational overhead to derive MaxTrust's max-flow-based tagger trustworthiness, using a 3.4 GHz P4 machine with 2GB memory running Debian 2.6.25. We repeated the measurement 5 times. The mean computation time to obtain the tagger trustworthiness using our max-flow-based method for all $200K$ users and for $T_{max} = 100$ is 629 sec. The computation time is almost independent of the number of trust seeds. The required memory is $\sim$ 550MB ($\sim$ 500MB for graph structure data, $\sim$ 50MB for computation).

We observe that the computation cost of our heuristic for submillion node graphs is not excessive. Using parallel computation techniques, such as MapReduce [Dean and Ghemawat 2004], as used for the computation of PageRank in Google's datacenters and by BotGraph [Zhao et al. 2009], we expect that this computation could scale to multimillion user social networks.

On the other hand, solving the optimal max-flow using the Edmonds-Karp algorithm is computationally prohibitive. For the same $200K$-user social graph, under the same machine configuration, Edmonds-Karp requires approximately 1 *million* sec.

7. RELATED WORK

Overview. Prior work has employed trust in social networks to improve system security [Danezis and Mittal 2009; Lesniewski-Laas and Kaashoek 2010; Post et al. 2011; Pujol and Delgado 2002; Ramachandran and Feamster 2008; Sovran et al. 2008; Tran et al. 2009; Yardi et al. 2008]. FaceTrust's main novelty lies in employing OSNs to provide lightweight, flexible, and relaxed identity attribute credentials. In addition, FaceTrust improves upon a max-flow-based trust inference method [Levien 2003] making it scalable with the number of trust seeds.

Social web of trust. The goal of FaceTrust is more related to the PGP Web of Trust (WoT) [Stallings 1995; Zimmerman 2009]. Like the PGP WoT, FaceTrust aims to circumvent the expensive and often monopolized Certificate Authorities to provide lightweight credentials. Unlike the PGP WoT, FaceTrust uses the intuitive OSN interface, and employs social tagging rather than key signing to derive trustworthiness. Furthermore, FaceTrust is easily extensible, and is not limited to certifying only public keys. Users can tag each other regarding multiple types of identity assertions, and the set of assertions can be extended by simply adding fields into a user's profile.

Birthday-paradox-based trust inference. SybilLimit [Yu et al. 2008] also exploits the fact that although attackers can create multiple Sybils, they are limited in their ability to create and sustain social acquaintances. SybilLimit performs special random walks of $O(\log |V|)$ length (called random routes) starting from trusted verifier nodes and a suspect node to determine whether the suspect is a Sybil. In a fast-mixing social graph, the last edge traversed by the random walk is drawn from the stationary distribution of the graph. Following from the generalized Birthday Paradox, the last edges of $\Theta(\sqrt{|E|})$ random walks from the verifier nodes and from the honest nodes intersect with high probability. The opposite holds if the suspect resides in a region of Sybil attackers connected with a disproportionally small number of edges to the honest node region.

In this case, the network has a higher mixing time and the last edges of random walks from the suspects are not drawn from the stationary distribution.

FaceTrust could employ SybilLimit instead of MaxTrust as follows. For each level of trust $0 \leq w \leq T_{max}$ we can prune the tagging similarity graph such that it includes only edges that denote greater than or equal to w similarity. We subsequently run SybilLimit for each user in the graph and for each level of trust and use as verifiers the trust seeds. The users (suspects) that are accepted for at most a trust level w are considered to have tagger credibility w. The reason we do not employ SybilLimit is that its computation cost would be $O(\sqrt{|E|}T_{max}|V| \log |V|)$, which is approximately $\sqrt{|E|}$ times more expensive than MaxTrust's under our sparse social graph setting.

Max-flow-based trust inference. Scalar max-flow-based trust inference computes the maximum flow over a trust graph from a trusted node (source) to a suspect node (sink) in order to determine whether the suspect is trustworthy. Levien and Aiken [1997] and Reiter and Stubblebine [1999] proposed scalar max-flow trust inference schemes for public key certification schemes such as the PGP WoT. They have also proved the resilience of maximum-flow-based trust metrics to node and edge attacks. In addition, Cheng and Friedman [2005] have shown that a node cannot increase its trust by creating Sybils, and needs to establish social edges with multiple honest nodes in order to attain the same trustworthiness as honest nodes.

We do not employ scalar trust inference because it is not sum-Sybilproof (Section 3.3.2). That is, they do not prevent an attacker from creating Sybils that obtain the same trust value as their creator. Thus, an attacker can increase the sum of the trust of the users he controls simply by adding Sybils.

Advogato [Levien 2003] and Sumup [Tran et al. 2009] use group max-flow-based trust inference toward a Sybil-resilient trust metric and a voter collection system, respectively. Group max-flow trust inference bounds the sum of the trust values of Sybils by the edge capacity of their creators. Sumup computes multiple-source maximum flow from the users to a single trusted vote collector with a DFS-based heuristic to decide which users can vote at least once. Although Sumup's DFS-based max-flow heuristic has comparable computation cost with MaxTrust's, it is designed to collect votes from a small fraction of users ($\leq 20\%$) in a social network. Thus, in our setting it can accept only a small fraction of honest users as trustworthy.

Bazzar [Post et al. 2011] also uses a max-flow-based technique to access the likely trustworthiness of users in online marketplaces. It uses the network formed from prior successful transactions as an input of the max-flow-based technique, thereby limiting trustworthiness manipulation. To reduce the computation cost, Bazzar uses a layered graph concept called multigraph, which contains a series of networks, where each subsequent network is a subgraph of the previous containing only those links with higher flows.

Eigenvector-based trust inference. In EigenTrust [Kamvar et al. 2003] and TrustRank [Gyöngyi et al. 2004] the node trust values are the left principal eigenvector e of the matrix c, where c_{ij} is the normalized pairwise trust between nodes i and j. Both schemes seed the computation of the eigenvector at a few selected trusted nodes. This computation expresses how trust flows among users through directed weighted edges. For a fast-mixing social graph TrustRank can be computed in $O(|V|log|V|)$ time and this value approximates the stationary distribution of the graph for users that reside in the honest fast-mixing region of the graph. Although for sparse and small world social graphs the computation cost of eigenvector-based trust inference is comparable to MaxTrust's, we do not employ it because Cheng and Friedman [2006] have shown that it is substantially manipulable under Sybil strategies.

Bayesian Sybil inference. Similar to MaxTrust, SybilInfer [Danezis and Mittal 2009] takes advantage of the fact that clusters of Sybils are connected to the honest regions of social networks with a disproportionally small number of edges. Its Bayesian Sybil detection method derives the probability of a suspect node being a Sybil, which is an explicitly actionable measure of trustworthiness. However, its computation cost is excessive for our setting ($O(|V|^2 \log |V|)$.

FaceTrust can also employ nonsocial-network-based Sybil defense techniques such as the ones proposed in Zhao et al. [2009] and Yu et al. [2009] to further limit the influence of Sybils in the system.

These techniques can be used in combination with FaceTrust to build more trustworthy and Sybil-resilient email, recommendation, and online review systems. BotGraph [Zhao et al. 2009] detects botnet spamming attacks that target Web email providers. BotGraph detects botnets by constructing large user-user graphs for links between email-exchanging users and looking for tightly connected subgraph components. As stated in Zhao et al. [2009] this technique is applicable in social graphs.

8. CONCLUSION

We presented FaceTrust, a system that leverages OSNs to provide lightweight, flexible, relaxed, and anonymous credentials. These credentials help users and services to assess the veracity of assertions made by online users. With FaceTrust, OSN users post identity assertions such as “Am I really 18 years old?” on their OSN profiles, and their friends explicitly tag these assertions as `true` or `false`. An OSN provider analyzes the social graph and the user tags to assess how credible these assertions are, and issues credentials annotated by veracity scores. Our analysis, real-world deployment, and simulation-based evaluation suggest that FaceTrust is effective in obtaining credible and otherwise unavailable identity information for online personas.

REFERENCES

Yong-Yeol Ahn, Seungyeop Han, Haewoon Kwak, Sue Moon, and Hawoong Jeong. 2007. Analysis of topological characteristics of huge online social networking services. In *Proceedings of the 16th International Conference on World Wide Web (WWW'07)*. 835–844.

Randy Baden, Neil Spring, and Bobby Bhattacharjee. 2009. Identifying close friends on the internet. In *Proceedings of the 8th ACM Workshop on Hot Topics on Networks (HotNets'09)*.

Leyla Bilge, Thorsten Strufe, Davide Balzarotti, and Engin Kirda. 2009. All your contacts are belong to us: Automated identity theft attacks on social networks. In *Proceedings of the 18th International Conference on World Wide Web (WWW'09)*.

Yazan Boshmaf, Ildar Muslukhov, Konsantint Beznosov, and Matei Ripeanu. 2011. The socialbot network: When bots socialize for fame and money. In *Proceedings of the 27th Annual Computer Security Applications Conference (ACSAC'11)*. 93–102.

Jan Camenisch and Els van Herreweghen. 2002. Design and implementation of the idemix anonymous credential system. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS'02)*. 21–30.

Qiang Cao, Michael Sirivianos, Xiaowei Yang, and Tiago Pregueiro. 2012. Aiding the detection of fake accounts in large scale social online services. In *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation (NSDI'12)*. 15.

Alice Cheng and Eric Friedman. 2005. Sybil-proof reputation mechanisms. In *Proceedings of the ACM SIGCOMM Workshop on Economics of Peer-to-Peer Systems (P2PEcon'05)*. 128–132.

Alice Cheng and Eric Friedman. 2006. Manipulability of pagerank under sybil strategies. In *Proceedings of the 1st Workshop on the Economics of Networked Systems (NetEcon'06)*.

George Danezis and Prateek Mittal. 2009. SybilInfer: Detecting sybil nodes using social networks. In *Proceedings of the 16th Annual Network and Distributed System Security Conference (NDSS'09)*.

Jeffrey Dean and Sanjay Ghemawat. 2004. MapReduce: Simplified data processing on large clusters. In *Proceedings of the 6th Conference on Symposium on Operating Systems Design and Implementation (OSDI'04)*. 10.

John R. Douceur. 2002. The sybil attack. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS'02)* Revised Papers. 251–260.

FaceTrust-Credentials. 2011. FaceTrust - Certify your identity through your online social network, web archive. http://web.archive.org/web/20111104214113/, http://www.facetrust.net/.

Minas Gjoka, Maciej Kurant, Carter T. Butts, and Athina Markopoulou. 2010. A walk in facebook: Uniform sampling of users in online social networks. In *Proceedings of the 29th Conference on Information Communications (INFOCOM'10)*. 2498–2506.

Ratan Kuman Guha, Ravi Kumar, Prabhakar Raghavan, and Andrew Tomkins. 2004. Propagation of trust and distrust. In *Proceedings of the 13th International Conference on World Wide Web (WWW'04)*. 403–412.

Zoltan Gyongyi, Hector Garcia-Molina, and Jan Pedersen. 2004. Combating web spam with trustrank. In *Proceedings of the 13th International Conference on Very Large Data Bases (VLDB'04)*. 576–587.

Amy Harmon. 2004. Amazon glitch unmasks war of reviewers. http://www.nytimes.com/2004/02/14/us/amazon-glitch-unmasks-war-of-reviewers.html.

Danesh Irani, Marco Balduzzi, Davide Balzarotti, Engin Kirda, and Calton Pu. 2011. Reverse social engineering attacks in online social networks. In *Proceedings of the 8th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA'11)*. 55–74.

Paul Jaccard. 1901. Etude comparative de la distribution florale dans une portion des alpes et des jura. *Bulletin del la Societe Vaudoise des Sciences Naturelles 37,* 547–579.

Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. 2003. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the 12th International Conference on World Wide Web (WWW'03)*. 640–651.

Adam Langley. 2013. Google security blog, enhancing digital certificate authority. http://googleonlinesecurity.blogspot.com/2013/01/enhancing-digital-certificate-security.html.

Jure Leskovec and Christos Faloutsos. 2006. Sampling from large graphs. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD'06)*. 631–636.

Chris Lesniewski-Laas and M. Frans Kaashoek. 2010. Whanau: A sybil-proof distributed hash table. In *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation (NSDI'10)*. 8.

Raph Levien. 2003. Attack-resistant trust metrics. www.levien.com/thesis/compact.pdf.

Raph Levien and Alexander Aiken. 1997. Attack-resistant trust metrics for public key certification. In *Proceedings of the 7th Conference on USENIX Security Symposium (SSYM'97)*. 18.

Merkin. 2006. Worth double the money. http://tinyurl.com/y8pqgvl.

Alan Mislove, Massimiliano Marcon, Krishna P. Gummadi, Peter Druschel, and Samrat Bhattacharjee. 2007. Measurement and analysis of online social networks. In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement (IMC'07)*. 29–42.

Marti Motoyama, Damon McCoy, Kirill Levchenko, Geoffrey M. Voelker, and Stefan Savage. 2011. Dirty jobs: The role of freelance labor in web service abuse. In *Proceedings of the 20th USENIX Security Symposium*.

Greg Norcie, Emilliano De Cristofaro, and Victoria Bellotti. 2013. Bootstrapping trust in online dating: Social verification of online dating profiles. In *Proceedings of the Financial Cryptography and Data Security Workshop on Usable Security (USEC'13)*.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The pagerank citation ranking: Bringing order to the web. Tech. rep., Stanford. http://www.cs.odu.edu/∼mln/teaching/cs791s07/?method=getElement&element=∼week4∼KleinVuppala.pdf.

Arlen Parsa. 2009. Belkin's amazon rep paying for fake online reviews. http://tinyurl.com/yzgp9co.

Radia Perlman. 1999. An overview of pki trust models. *IEEE Netw. 13,* 6, 38–43.

Ansley Post, Vijit Shah, and Alan Mislove. 2011. Bazaar: Strengthening user reputations in online marketplaces. In *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation (NSDI'11)*. 14.

Josep M. Pujol and Ramon Sangesa Jordi Delgado. 2002. Extracting reputation in multi agent systems by means of social network topology. In *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'02)*. 467–474.

Anirudh Ramachandran and Nick Feamster. 2008. Authenticated out-of-band communication over social links. In *Proceedings of the 1st Workshop on Online Social Networks (WOSN'08)*. 61–66.

Venugoplalan Ramasubramanian and Emin Gun Sirer. 2005. Perils of transitive trust in the domain name system. In *Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement (IMC'05)*. 35.

Michael Reiter and Stuart Stubblebine. 1997. Toward acceptable metrics of authentication. In *Proceedings of the IEEE Symposium on Security and Privacy (SP'97)*. 10–20.

Michael Reiter and Stuart Stubblebine. 1999. Authentication metric analysis and design. *ACM Trans. Inf. Syst. Secur. 2*, 2, 138–158.

Rental-Scam. 2013. Waterloo regional police service, apartment rental scam. http://www.wrps.on.ca/staying-safe/fraud-prevention/apartment-rental-scam.

Oliver Richters and Tiago P. Peixoto. 2011. Trust transitivity in social networks. *PLoS One 6,* 4.

Michael Sirivianos, Kyungbaek Kim, and Xiaowei Yang. 2009. FaceTrust: Assessing the credibility of online personas via social networks. In *Proceedings of the 4th USENIX Conference on Hot Topics in Security (HotSec'09)*. 2.

Yair Sovran, Alana Libonati, and Jinyang Li. 2008. Pass it on: Social networks stymie censors. In *Proceedings of the 7th International Conference on Peer-to-Peer Systems (IPTPS'08)*. 3.

William Stallings. 1995. *Protect Your Privacy: A Guide for PGP Users*. Prentice-Hall.

Dinh Nguyen Tran, Bonan Min, Jinyang Li, and Lakshminarayanan Subramanian. 2009. Sybil-resilient online content rating. In *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation (NSDI'09)*. 15–28.

Kevin Walsh and Emin Gun Sirer. 2006. Experience with an object reputation system for peer-to-peer filesharing. In *Proceedings of the 3rd Conference on Networked Systems Design and Implementation (NSDI'06)*.

Alma Whitten and Doug Tygar. 1999. Why johnny can't encrypt: A usability evaluation of pgp 5.0. In *Proceedings of the 8th Conference on USENIX Security Symposium (SSYM'99)*. 14.

Wikipedia. 2013. Root certificate. http://en.wikipedia.org/wiki/Rootcertificate.

Baoning Wu, Vinay Goel, and Brian D. Davison. 2006. Topical trustrank: Using topicality to combat web spam. In *Proceedings of the 15th International Conference on World Wide Web (WWW'06)*. 63–72.

Zhi Yang, Chrito Wilson, Xiao Wang, Tingting Gao, Ben Y. Zhao, and Yafei Dai. 2011. Uncovering social network sybils in the wild. In *Proceedings of the ACM SIGCOMM Conference on Internet Measurement Conference (IMC'11)*. 259–268.

Sarita Yardi, Nick Feamster, and Amy Bruckman. 2008. Photo-based authentication using social networks. In *Proceedings of the 1st Workshop on Online Social Networks (WOSN'08)*. 55–60.

Haifeng Yu, Michael Kaminsky, Phillip B. Gibbons, and Abraham Flaxman. 2006. SybilGuard: Defending against sybil attacks via social networks. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM'06)*. 267–278.

Haifeng Yu, Phillip Gibbons, Michael Kaminsky, and Feng Xiao. 2008. A near-optimal social network defense against sybil attacks. In *Proceedings of the IEEE Symposium on the 29th IEEE Symposium on Security and Privacy (SP'08)*. 3–17.

Haifeng Yu, Chenwei Shi, Michael Kaminsky, Phillip B. Gibbons, and Feng Xiao. 2009. DSybil: Optimal sybil-resistance for recommendation systems. In *Proceedings of the 30th IEEE Symposium on Security and Privacy (SP'09)*. 283–298.

Jian Zhang, Phillip Porrs, and Johannes Ullrich. 2008. Highly predictive blacklisting. In *Proceedings of the 17th Conference on Security Symposium (SS'08)*. 107–122.

Yao Zhao, Yinglian Xie, Fang Yu, Qifa Ke, Yuan Yu, Yan Chen, and Elliot Gillum. 2009. Botgraph: Large scale spamming botnet detection. In *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation (NSDI'09)*. 321–334.

Philip R. Zimmerman. 1995. *The Official PGP User's Guide*. MIT Press.

Received July 2012; revised September 2013; accepted November 2013