Assessing the Veracity of Identity Assertions via OSNs

Michael Sirivianos
Telefonica Research
msirivi@tid.es

Kyungbaek Kim
UC Irvine
kyungbak@uci.edu

Jian Wei Gan
TellApart
wei@tellapart.com

Xiaowei Yang
Duke University
xwy@cs.duke.edu

***Abstract*—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 strongly correlate with the ground truth, even when a large fraction of the social network users is dishonest and employs the Sybil attack.**

I. 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 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" [4] from a user claiming "I was a chef for many years." Should she believe it, given that users have been caught writing biased positive reviews for their own books [6] or their company's products [1]?

Real-world remedies to this problem typically forgo a user's anonymity. Moreover, verifying a user's identity can be costly and time consuming. For instance, Amazon provides a "Real Name" badge to a user that wishes to sign his posts by his real name [5]. It verifies a user's name using his credit card information.

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 [3]), and apply techniques such as *idemix* [9] to make the certificates anonymous, unlinkable, and non-transferable. However, this approach involves centralized manual verification, and could be a financial [3] as well as a usability burden on users [31].

We propose FaceTrust, a system that enables online personas to cheaply obtain credentials that indicate the veracity of their identity statements without sacrificing their anonymity. Our insight is that in many settings, online users or services do not require strong authentication, and can benefit greatly from likely-to-be-true identity information. For example, a user may only need to know that a reviewer's declared profession has a high chance of being authentic. Similarly, it may suffice for an adult site to know that a user's age information is likely to be true.

FaceTrust mines and enriches information embedded in online social networks (OSNs) to provide lightweight and flexible digital credentials for the identity assertions. We observe that OSNs already enable users to express trust towards their friends in the form of friend connections and lists (or circles). We propose to extend this limited ability by allowing users to declare whether they consider the identity assertions of their friends credible (§III-A).

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, *e.g.*, "Am I really over 18?" Since the identity information on the OSN profile is inserted by the user, the OSN provider can not directly infer the veracity of the information. Thus, the OSN let the user ask his friends to respond to this poll by tagging his assertion as `true` or `false`. Based on the tagging information, the OSN employs a veracity scoring mechanism (§III-C2) to estimate a score that reflects how credible an assertion is. We call this score *assertion veracity* (§III-B). The scoring mechanism also employs transitive trust inference [15], which needs to be attack-resistant because users may post false assertions, tag incorrectly or lie, and create Sybil accounts [13], [7].

The intuition behind our trust inference scheme is that benign (honest) users tend to tag correctly and similarly. We compare a user's tags with those from his friends on the set of assertions they both tag, and use the similarity between the tags as pairwise trust values on the social graph edges. We then compute a Sybil-resistant *tagger trustworthiness* score for each user, using our proposed max-flow-based scoring mechanism that is more scalable than existing approaches. Our scheme seeds trust at pre-selected 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, we derive an assertion's veracity 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}` (§III-E). Verifiers (online services or human users) can use this OSN-issued credential to regulate their interactions with the user that posted the assertion. For usability, our context-specific credential scheme allows a user to certify his online assertions with a web interface without involving user-side cryptography (§III-E).

To evaluate FaceTrust we have built and deployed a Facebook application, which has amassed over 1000 users. We have also performed simulations on a 200K-user sample of a crawled social graph (§IV). We 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

978-1-4673-0298-2/12/$31.00 © 2012 IEEE

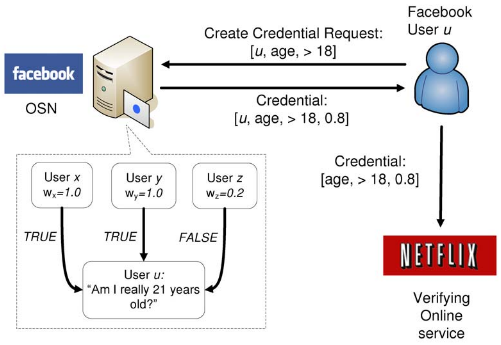


Fig. 1: **FaceTrust overview and an age verification example.**

employs the Sybil attack.

The rest of the paper is organized as follows: §II describe the overview of FaceTrust with an example and discuss FaceTrust ś assumptions and goals. §III provides the design of FaceTrust in detail. §IV presents an evaluation of FaceTrust . §V provides related works and §VI concludes.

II. 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.

A. An Example

We first use an age-verification example to shed light on how FaceTrust roles interact. As shown in Figure 1, user u attempts to access an age-restricted movie at the Netflix website. 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. His friends must have also tagged 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. Since not all users are equally trustworthy, the OSN provider has computed a trustworthiness score (w) for each tagger x, y, and z by analyzing the social graph and their tagging history as we describe in §III-C. 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 (§III-B). Subsequently, the OSN issues an age credential with a veracity score that certifies that the user belongs to the restricted age group.

In addition to age verification, we envision that FaceTrust credentials will benefit Internet users and online services in many other ways. More examples include but are not limited: a) assessing the authority or relevance of online reviews or wikipedia articles with profession credentials; and b) verifying participant eligibility in online surveys or citizen journalism sites [2] with credentials for age, location or other identity attributes.

B. Assumptions

In designing FaceTrust, we make the following assumptions:

Users carefully vet FaceTrust friend requests: We rely on the assumption that establishing friend connections in the FaceTrust social network is a resource-intensive task. Users can use common-secret-based techniques to verify that a OSN friend request originates from a real acquaintance and not an imposter [8]. We also assume that a user selects as OSN friends users that will not try to harm him by tagging his honest assertions as `false`.

Trusted credential issuing authority: We assume that the OSN provider reliably issues credentials based on the input of its users. We also assume that the OSN provider does not reveal a user's tags to others. Furthermore, users may wish to remain anonymous and untraceable by the verifiers. We assume that the OSN provider protects the privacy of its users by not revealing their identity and the list of online services or users that verify its users' credentials. Also, we assume that users employ common sense to make sure that the identity credentials do not identify them uniquely.

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, since an assertion in FaceTrust is about ground truth (e.g. age, profession, sex and etc.) rather than personal taste. We validate this assumption to some extent in §IV-B. We treat users that consistently tag mistakenly not due to malicious intent, but due to lack of knowledge, as *dishonest* (§II-C).

C. Threat Model

FaceTrust's design copes with the following threats by malicious users aiming 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 [13] 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`.

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 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 appear credible.

Camouflage attack: This threat model resembles what Kamvar et al. [18] refer to as "malicious nodes with camouflage." 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. 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.

D. Goals

FaceTrust's design is driven by the following goals:

Attack-resistant: It should be difficult for false assertions to appear trustworthy by having high veracity. Although our design is attack-mitigating, it cannot ensure the correctness of the veracity scores in the presence of devoted adversaries. Thus, it 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. That is, FaceTrust provide *relaxed* credentials rather than the truthfulness of a statement.

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, *e.g.*, 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, *i.e.*, the verifier should be assured that a credential is issued by a trusted authority. They should satisfy integrity, *i.e.*, the assertion, veracity and context fields should be inalterable once the credential is issued. 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. Finally, the credentials should preserve anonymity.

III. Design

A. Social Tagging

FaceTrust uses *social tagging* to obtain credible identity information of online users. By social tagging, we refer to OSN users posting assertions about their attributes and their friends tagging them as `true` or `false`. Our approach is similar to the PGP Web of Trust [35], in that we allow only friends of a user to tag (certify) the user's assertion. Our rationale is two-fold. First, most of the assertions posted by a user can only be reliably evaluated by people who know him (friends). Second, since 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]`, *e.g.*, `[> 18]` means that the user claims to be older than 18. We use distinct types because a user's tendency to correctly tag assertions may vary by type (§III-B), and to address the camouflage attack (§III-C1).

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. 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. Since the tags represent sensitive information, they are only known to the OSN provider and their tagger.

Users post assertions on their OSN profile and tag their friends' assertions using our "Am I Really?" (AIR) Facebook application: http://apps.facebook.com/am-i-really. AIR employs a "game with a purpose" design to incentivize social tagging.

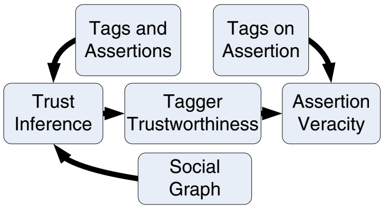


Fig. 2: **Combining social tagging with trust inference to derive the veracity of user assertions.**

B. 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 to make them appear true, 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, i.e., an assertion with higher veracity than another is more likely to be true.

As shown in Figure 2, the inputs for computing an assertion's veracity are the tags on the assertion and their 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 §III-C2. We then weigh an assertion's tags 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. Equation 1 ensure that the sum of the weights w_j^t of `true` tags should be more thean 0.75 of the sum of the weights of all users in F_i for a_A to be more than 0.5 veracity. Thus, this design is biased towards making it difficult for users to make flase assertions appear true. However, malicious users may abuse this design to make `true` assertions of a user non-credible. 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.

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

C. 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, i.e., 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 select 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 §III-C1 and §III-C2.

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 [17]. 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 use special assertions for each type - "Do I honestly tag the <type> assertions of my friends?" - to derive user-defined similarity us_{ij}^t by a user i on his friend j for type t. 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 that users i and j have tagged: the larger N_t is, the higher a should be. When N_t is large, we presume that $hs_{ij}^t(C_t/N_t)$ 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 (S-shaped) function : $a = (1 + e^{b-N_t})^{-1}$. b is a small constant, and we set $b = 5$ in this paper.

We then transform the social graph into a trust graph by assigning the tagging similarity ts_{ij}^t 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 (§II-C). 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.

2) 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 group max-flow-based class of trust inference algorithms [21], [30], which have been shown to be sum-Sybilproof [21], i.e., 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 (trusted seeds) and propagates to the other users through the trust graph. A seed is a highly trusted user, *e.g.*, 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 trusted 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 trusted 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* [21], 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 trusted seeds and honest users."

In addition, the selection of the trusted seeds and the number of trusted seeds is paramount to the attack resilience of the system. This is because an attacker that manages to be friend trusted seeds and to build up high tagging similarity with them can greatly manipulate trust assignment. When the trust inference method employs numerous trusted 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 trusted seed can be a difficult task for attackers.

Hence, one desirable feature of trust inference methods is to be efficiently computable for numerous trusted seeds. To this end, the method's computation cost should be mostly independent of the number of trusted 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 also note that determining which users in a social graph can be designated as trusted seeds is an important challenge. Gyongyi et al. [16] addressed this challenge in the context of TrustRank, an eigenvector-based trust inference method for web pages. Their solution also applies in our setting. More recently, Wu et al. proposed improvements over the seed selection algorithm [32] introducing topical TrustRank.

Our method is inspired by the Advogato [21] 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.

Compared to Advogato, MaxTrust's advantage is that although it employs multiple trusted seeds from a set $S \subset V$, it does not need to be run for each seed. Instead in a single

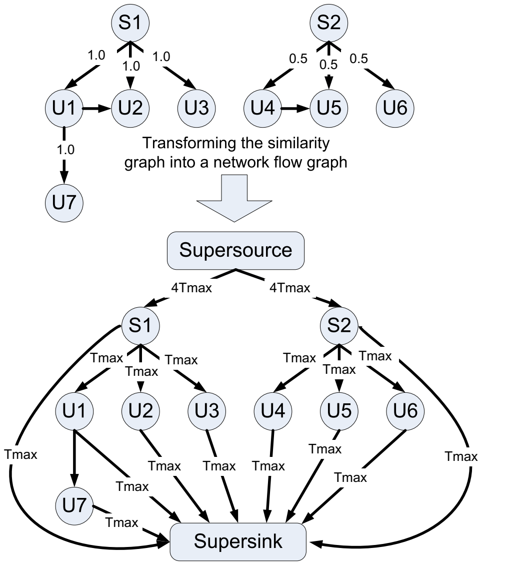


Fig. 3: **The tagging similarity-based trust graph and its conversion into a MaxTrust network flow graph. The capacity $C_{supersource}$ in this example is $8 \cdot T_{max}$. MaxTrust results in all users except $U7$ having tagger trustworthiness equal to T_{max}.**

run (max-flow computation), it considers all the trusted seeds. This results in MaxTrust being $\Theta(|S|)$ times more efficient than Advogato.

MaxTrust computes the tagger trustworthiness $0 \leq w_i^t \leq T_{max}$ using a 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 the following two phases:

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 3). We then add an edge from the supersource to each trusted user $s \in S$. We add a directed edge from each user, except of 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: $C_{uv} = (C_u - T_{max}) \dfrac{ts_{uv}}{\sum_{z \in F_u} ts_{uz}}$, 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 II-C.

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 $O(T_{max}(1 - p_d)|V||E|)$, since it takes at most $C_{supersource} = T_{max}(1 - p_d)|V|$ augmentations. This is computationally prohibitive [28], 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 $parent(j)$.

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. 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 [23]).

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 $j \rightarrow supersink$, 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 $j \rightarrow supersink$. 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.

For an analysis of the assurances provided by the max-flow-based tagger trustworthiness mechanism, see Section 4.2 of [28].

D. Mitigating Sybil Assertion Posters

We now describe how we improve the above scheme to defend against the Sybil assertion poster attack (§II-C). Since 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 group of colluding 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 falsified 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 group of colluding users creates multiple Sybil accounts to overcome the credential quotas. 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.

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

$\overline{w}$ is the tagger trustworthiness value for which $(1-p_d)|V|$ users have great 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, that assigns a minimum veracity $a_A \cdot c$ to assertion A in case the tagger trustworthiness w^t_j of j is 0.

Since our trust inference method assigns very low tagger trustworthiness scores to multiple Sybils and less tagger trustworthiness to dishonest than to honest users, this adjustment results in decreased veracity for assertions posted by colluders in this attack.

E. OSN-Issued Credentials

After the OSN provider (§II-B) obtains the assertion veracity score for a user i's assertion A^t_i, it can issue a web-based relaxed credential for this assertion, when the need arises. As shown in Figure 1, a credential issued by an OSN will include the assertion type t, the assertion A^t_i, and the assertion veracity score.

We use non-cryptographic credentials that satisfy the goals listed in §II-D. Each credential as seen by the verifiers comprises:

- 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, 1.0, 17 tags]
- This is a great textbook and I highly recommend it ...
- http://www.amazon.com/review/...

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.

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.

IV. 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 the first two aspects in turn, and refer the reader to Section 6.3 of [28], where we illustrate our scheme's computational feasibility.

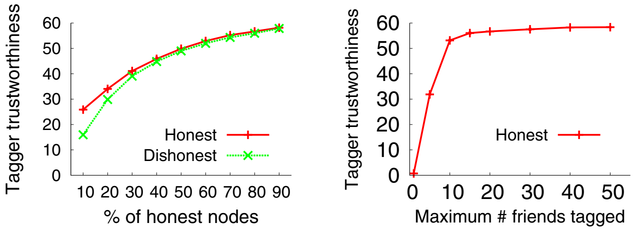
A. Effectiveness

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, i.e., under which conditions and attack strategies false assertions can obtain high veracity.

For a more realistic evaluation, we use a crawled sample of the Facebook social graph [14], which consists of a $200K$-user connected component obtained from a $1M$-user sample via the "forest fire" sampling method [19]. The average and maximum number of friends of each user in the graph is ~ 24 and 313, respectively. The diameter of this graph is 18 and the clustering coefficient is 0.159.

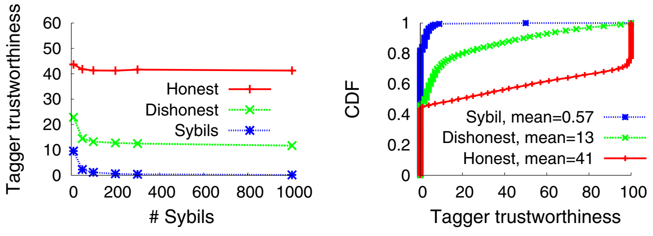
1) General Simulation Settings: Each user in the social graph posts a single assertion of the same type on his profile. Honest users always post true assertions and dishonest users always post false assertions. Furthermore, the honest users tag as `true` the assertions posted by their honest friends 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



(a) (b)

Fig. 4: **Mean tagger trustworthiness: (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.**



(a) (b)

Fig. 5: **(a) Mean tagger trustworthiness as a function of the number of Sybils each dishonest user creates; (b) CDF of the tagger trustworthiness of honest, dishonest and Sybil users; Dishonest users employs 200 Sybils each, $F = 20$ and 50% of users are honest.**

increase their tagging similarity with trustworthy users. This is a manifestation of the tagger camouflage attack (§II-C).

Both honest and dishonest users are randomly distributed in the social graph. The case of dishonest colluders that forms a group in the social graph is discussed in §IV-A3. 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.

We obtain the tagger trustworthiness as described in §III-C2. We do not consider the user-defined similarity (§III-C1), as we use no notion of a priori trust between users. We set $T_{max} = 100$ (§III-C2).

For each experiment, the minimum sum of the trustworthiness of taggers M (§III-B) is equal to the average tagger trustworthiness of honest users. We set $c = 0.2$ (§III-D). We employ 1000 trusted seeds, which are randomly selected among the honest users. We repeat each experiment 5 times and plot the mean.

2) Tagger Trustworthiness Effectiveness: As described in §III-B, the tags on assertions are weighted by their tagger's trustworthiness. Therefore, we first need to examine the effectiveness of tagger trustworthiness (§III-C) 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 4(a), we observe that the trustworthiness of honest users is substantially higher than the one of dishonest users when the portion of honest users is small. This holds despite the fact that honest and dishonest users have the same connectivity in the social graph.

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. Since tagging similarity captures the difference in tagging behavior between dishonest and honest users, this translates to low pairwise trust between them. In addition, since trust is seeded at honest users, MaxTrust's transitive trust mechanism assigns lower tagger trustworthiness to dishonest users. This result demonstrates the importance of tagging similarity, which results in dishonest users having less influence on the system's operation.

Figure 4(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 (§III-C1) 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.

Dishonest users employ Sybil Taggers: To evaluate the scheme'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 5(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 (§III-C2), which limits the amount of trust that can be assigned to the Sybils of a dishonest user.

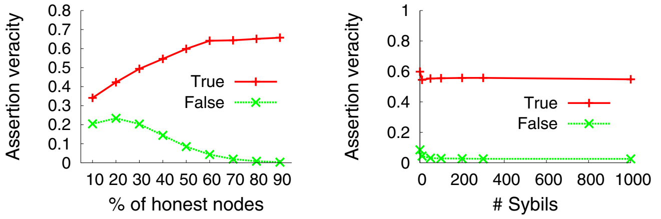
We now examine how tagger trustworthiness is distributed among the users. Figure 5(b) depicts the CDF of the tagger trustworthiness of honest, dishonest and Sybil users. As can be seen, there is substantial variance in the trustworthiness scores of honest and dishonest taggers. Nevertheless, on average dishonest users have substantially lower trustworthiness due to their decreased tagging similarity with honest users. Furthermore, almost 80% of Sybil users has 0 tagger trustworthiness.

In [28] we further evaluate our system under the assertion poster camouflage attack (§II-C).

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 (§III-B) under varying attack scenarios.

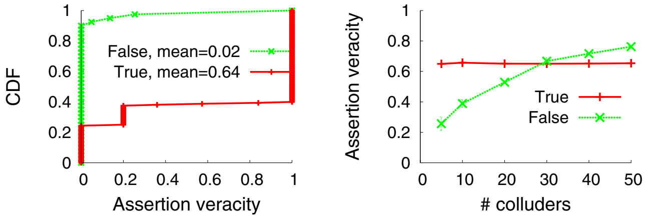
Dishonest users do not employ Sybils: Figure 6(a) plots the mean veracity of honest and false assertions as a function of the portion of the users that are honest, when dishonest nodes do not employ Sybils. We observe that when the fraction of honest users exceeds 50%, the mean veracity of true assertions substantially exceeds that of false ones. Unlike plain majority voting, our mechanism assigns low veracity to false assertions even when the fraction of dishonest users is large. This is because MaxTrust assigns lower tagger trustworthiness to dishonest users and their tags are discounted.

Dishonest users employ Sybil taggers: Figure 6(b) shows the veracity of true and false assertions when dishonest users



(a) (b)

Fig. 6: **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.**



(a) (b)

Fig. 7: **(a) CDF of assertion veracity when dishonest users employ Sybils; b) mean veracity of assertions posted by dishonest colluders as a function of the colluder group size; 80% of users are honest and $F = 20$.**

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 (Equation 2).

Figure 7(a) shows how veracity is distributed among true and false assertions. 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. 24.3% of true assertions obtain 0 veracity. The true assertions with $c = 0.2$ veracity belong to honest users with 0 tagger trustworthiness (Equation 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), i.e., increasing the adoption of social tagging. Incorrectly assessed assertions can be further avoided by designating more trusted 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.

Dishonest focused colluders: We also evaluate the case in which dishonest users form colluders groups. 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 7(b) depicts the mean veracity of the assertions posted by the dishonest users as a function of the size of the colluder groups.

We observe that the false assertions of colluders can get higher average veracity than the true assertions only if the colluder group size exceeds a relatively high threshold (30). 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 trusted 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 trustworthiness become more prominent and the assertions of colluders get high veracity.

This result reveals 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, which may discourage them from orchestrating an attack.

Dishonest users employ Sybil assertion posters: We now evaluate our system when a group of dishonest users performs the Sybil assertion poster attack (§II-C). Each group of colluders creates Sybils to which all the colluders connect to. 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.

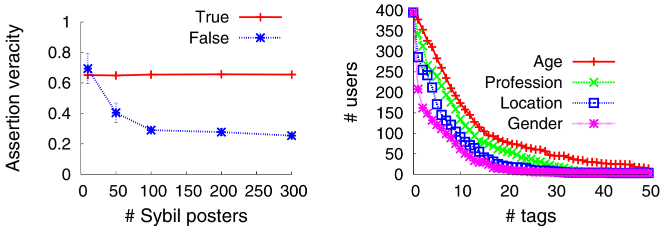
In Figure 8(a), when the number of Sybil assertion posters is small, e.g., 10, we observe that the assertion veracity is high. Since the number of Sybils is small, MaxTrust does not assign low tagger trustworthiness to them. Consequently, Equation 3 (§III-D) does not mitigate this attack, because both 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 (§III-D) on the number of credentials each user can request.

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 8(a). This result indicates the importance of multiplying the assertion veracity by the poster's tagger trustworthiness as described in §III-D. Furthermore, rational dishonest users incur a cost to create Sybils, e.g., solving CAPTCHAs during Facebook account registration, which further limits this attack.

B. Facebook Deployment

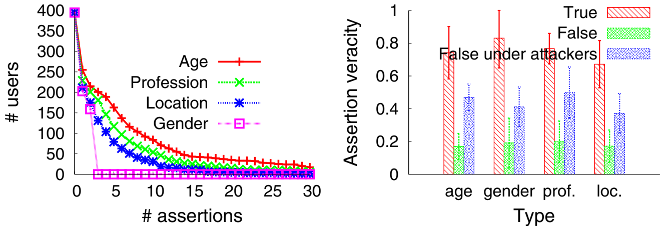
FaceTrust requires a new form of user input: assertions and tags. In addition, in order for the veracity scores to correlate positively with the ground truth, it requires trustworthy users to tag honestly and similarly. These facts motivate us to ask: Are users willing to tag their friends' tags? How often and honestly will they tag? To answer these questions, we deployed the "Am I Really?" (AIR) Facebook application (§III-A) 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 data set consisting of 1108 real Facebook users. (Duke University IRB Protocol 3015.) 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



(a) (b)

Fig. 8: **a) 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; b) CCDF of the number of "Am I Really?" users as a function of the number of tags per user for each assertion type.**



(a) (b)

Fig. 9: **a) CCDF of the number of users as a function of the number of posted assertions per user for each assertion type; b) 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.**

concerning those 395 users, since they are the only ones that can tag friends in AIR. Our data set includes 2410 social connections established between Sept. 1st, 2009 and Jan. 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 trusted seeds, set $T_{max} = 10$ and assume that 90% of the network consists of honest users. We incorporate user-defined similarity (§III-C1) in the computation of tagging similarity, using $b = 5$. We again set $c = 0.2$ (§III-D).

Figure 8(b) 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 users have tagged at least 8, 6, 4, and 1 time for 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, the number needed to obtain accurate tagger trustworthiness (Figure 4(b) in §IV-A2).

Figure 9(a) 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 now examine the AIR profiles of 10 out of the 395 users, for which we know 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 profession assertions, and 10 false gender assertions. Each of these assertions were tagged ~ 6 times on average by distinct users.

Figure 9(b) 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 of false assertions when we inject 20 dishonest users in AIR's social graph. The injected dishonest users do not represent real Facebook accounts. They are connected to the 10 real honest users, such that each of these real users is AIR-friends with two distinct dishonest users. 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 9(b), 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. 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.

V. Related Work

Overview: Prior work has employed trust in social networks to improve system security [25], [29], [26], [20], [33], [30], [12], [24]. 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 [21] making it scalable with the number of trusted seeds.

Social web of trust: The goal of FaceTrust is more related to the PGP Web of Trust (WoT) [35]. 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 [34] 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.

FaceTrust could employ SybilLimit instead of MaxTrust, however 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 et al. [22] and Reiter et al. [27] 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 et al. [10] have shown that a node cannot increase its trust by creating Sybils. We do not employ scalar trust inference because it is not sum-Sybilproof (§III-C2).

Advogato [21] and Sumup [30] 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 [24] 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 manifulation. To reduce the computation cost, Bazzar uses a layered graph concept called multi-graph, 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 [18], TrustRank [16] and Personalized PageRank [36], 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. Although, for sparse and small-world social graphs the computation cost of the above eigenvector-based trust inference methods is comparable to MaxTrust's, we do not employ them because Cheng et al. [11] have shown that it is substantially manipulable under Sybil strategies.

Bayesian Sybil inference: Similar to MaxTrust, SybilInfer [12] 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 ($O(|V|^2 \log|V|)$).

VI. 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.

VII. Acknowledgements

This work was funded in part by NSF CAREER Award CNS-0845858, Award CNS-1017858 and Award CNS-0925472.

References

[1] Belkin's Amazon Rep Paying For Fake Online Reviews. http://tinyurl.com/yzgp9co.

[2] Unedited. Unfiltered. News. iReport.com. www.ireport.com.

[3] Verisign: Personal Digital Certificate Enrollment. https://personalid.verisign.com.au.

[4] Worth DOUBLE the money. http://tinyurl.com/y8pqgvl.

[5] Your Real Name ^{TM}Attribution. http://www.amazon.com/gp/help/customer/display.html?nodeId=14279641.

[6] Amazon glitch outs authors reviewing own books. http://www.ctv.ca/servlet/ArticleNews/story/CTVNews/1076990577460_35, 2004.

[7] So, Why Does the Air Force Want Hundreds of Fake Online Identities on Social Media? http://bit.ly/g8vDhZ, 2011.

[8] R. Baden, N. Spring, and B. Bhattacharjee. Identifying Close Friends on the Internet. In *HotNets*, 2009.

[9] J. Camenisch and E. V. Herreweghen. Design and Implementation of the idemix Anonymous Credential System. In *ACM CCS*, 2002.

[10] A. Cheng and E. Friedman. Sybilproof Reputation Mechanisms. In *P2PEcon*, 2005.

[11] A. Cheng and E. Friedman. Manipulability of PageRank under Sybil Strategies. In *NetEcon*, 2006.

[12] G. Danezis and P. Mittal. SybilInfer: Detecting Sybil Nodes using Social Networks. In *NDSS*, 2009.

[13] J. R. Douceur. The Sybil Attack. In *IPTPS*, March 2002.

[14] M. Gjoka, M. Kurant, C. T. Butts, and A. Markopoulou. A Walk in Facebook: Uniform Sampling of Users in Online Social Networks. In *INFOCOM*, 2010.

[15] R. K. Guha, R. Kumar, P. Raghavan, and A. Tomkins. Propagation of Trust and Distrust. In *WWW*, 2004.

[16] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen. Combating Web Spam with TrustRank. In *VLDB*, 2004.

[17] P. Jaccard. Etude Comparative de la Distribution Florale dans une Portion des Alpes et des Jura. In *Bulletin del la Societe Vaudoise des Sciences Naturelles 37, 547-579*, 1901.

[18] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The EigenTrust Algorithm for Reputation Management in P2P Networks. In *WWW*, 2003.

[19] J. Leskovec and C. Faloutsos. Sampling from Large Graphs. In *SIGKDD*, 2006.

[20] C. Lesniewski-Laas and M. F. Kaashoek. Whanau: A Sybil-proof Distributed Hash Table. In *NSDI*, 2010.

[21] R. Levien. Attack-resistant Trust Metrics. http://www.levien.com/thesis/compact.pdf, 2003.

[22] R. Levien and A. Aiken. Attack-resistant Trust Metrics for Public Key Certification. In *Usenix Security*, 1997.

[23] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and S. Bhattacharjee. Measurement and Analysis of Online Social Networks. In *IMC*, 2007.

[24] A. Post, V. Shah, and A. Mislove. Bazaar: Strengthening user reputations in online marketplaces. In *NSDI*, 2011.

[25] J. M. Pujol and R. S. J. Delgado. Extracting Reputation in Multi Agent Systems by Means of Social Network Topology. In *AAMAS*, 2002.

[26] A. Ramachandran and N. Feamster. Authenticated Out-of-Band Communication Over Social Links. In *WOSN*, 2008.

[27] M. Reiter and S. Stubblebine. Authentication Metric Analysis and Design. In *ACM TISSEC*, 1999.

[28] M. Sirivianos, K. Kim, J. W. Gan, and X. Yang. Assessing the Veracity of Online Identity Assertions via OSNs. http:// www.cs.duke.edu/m̃sirivia/publications/facetrust-tech-report-comsnets.pdf, 2010.

[29] Y. Sovran, A. Libonati, and J. Li. Pass it on: Social Networks Stymie Censors. In *IPTPS*, 2008.

[30] D. N. Tran, B. Min, J. Li, and L. Subramanian. Sybil-Resilient Online Content Rating. In *NSDI*, 2009.

[31] A. Whitten and D. Tygar. Why Johnny can't Encrypt: A Usability Evaluation of PGP 5.0. In *USENIX Security*, 1999.

[32] B. Wu, G. V., and D. B. D. Topical TrustRank: Using Topicality to Combat Web Spam. In *WWW*, 2006.

[33] S. Yardi, N. Feamster, and A. Bruckman. Photo-Based Authentication Using Social Networks. In *WOSN*, 2008.

[34] H. Yu, P. Gibbons, M. Kaminsky, and F. Xiao. A Near-Optimal Social Network Defense Against Sybil Attacks. In *IEEE S&P*, 2008.

[35] P. R. Zimmerman. The Official PGP User's Guide. In *MIT Press*, 1995.

[36] G. Jeh, J. Widom. Scaling Personalized Web Search. In *WWW*, 2003.