Strength in Numbers: Robust Tamper Detection in Crowd Computations

Bimal Viswanath
MPI-SWS
bviswana@mpi-sws.org

Muhammad Ahmad Bashir
Northeastern University
ahmad@ccs.neu.edu

Muhammad Bilal Zafar
MPI-SWS
mzafar@mpi-sws.org

Simon Bouget
IRISA/INRIA Rennes
simon.bouget@irisa.fr

Saikat Guha
Microsoft Research India
saikat@microsoft.com

Krishna P. Gummadi
MPI-SWS
gummadi@mpi-sws.org

Aniket Kate
Purdue University
aniket@purdue.edu

Alan Mislove
Northeastern University
amislove@ccs.neu.edu

ABSTRACT

Popular social and e-commerce sites increasingly rely on *crowd computing* to rate and rank content, users, products and businesses. Today, attackers who create fake (Sybil) identities can easily tamper with these computations. Existing defenses that largely focus on detecting individual Sybil identities have a fundamental limitation: Adaptive attackers can create hard-to-detect Sybil identities to tamper arbitrary crowd computations.

In this paper, we propose Stamper, an approach for detecting tampered crowd computations that significantly raises the bar for evasion by adaptive attackers. Stamper design is based on two key insights: First, Sybil attack detection gains *strength in numbers*: we propose statistical analysis techniques that can determine if a large crowd computation has been tampered by Sybils, even when it is fundamentally hard to infer *which* of the participating identities are Sybil. Second, Sybil identities *cannot forge the timestamps of their activities* as they are recorded by system operators; Stamper analyzes these unforgeable timestamps to foil adaptive attackers. We applied Stamper to detect tampered computations in Yelp and Twitter. We not only detected previously known tampered computations with high accuracy, but also uncovered tens of thousands of previously unknown tampered computations in these systems.

Categories and Subject Descriptors

J.4 [**Computer Applications**]: Social and Behavioral Sciences; K.6 [**Management of Computing and Information Systems**]: Security and Protection

COSN'15, November 2–3, 2015, Palo Alto, California, USA.

ACM 978-1-4503-3951-3/15/11 ...$15.00.
DOI: http://dx.doi.org/10.1145/2817946.2817964.

General Terms

Security, Design, Algorithms, Measurement

Keywords

Sybil attacks; crowd computing; social networks; Twitter; Yelp

1. INTRODUCTION

Popular social networking and e-commerce sites are increasingly employing *crowd computing* to rate and rank content, users, products, and businesses. In such systems, crowd computations involve polling the "wisdom" or "opinions" of crowds—the users of the system—to provide a variety of recommendation services to their customers. For example, social networking and media sites like Facebook, Twitter, YouTube, and Reddit recommend content (be it business pages, news stories, videos, photos, or web pages) based on the number of users who posted, endorsed, or liked the content. Similarly, e-commerce sites like Amazon and eBay rely on their users to rate and review products and sellers. Some online sites like Yelp and TripAdvisor are dedicated to crowd-sourcing the rating of businesses.

In many crowd computation systems, users operate behind easy-to-create *weak* identities and consequently, they are vulnerable to Sybil attacks [18], where an attacker creates multiple fake identities with the goal of manipulating the aggregate opinion of the crowd. There are thriving underground markets for launching such tampering attacks on the crowd-sourcing sites mentioned above [27, 32, 35]; typically, the more popular a site, the greater the frequency and magnitude of such attacks. Existing Sybil defenses have mostly taken the approach of detecting individual Sybils [15, 36, 37, 41], enabling the operator to either suspend the Sybils or nullify their contribution to the crowd computation.

In this paper, we begin by highlighting a *fundamental limitation* of defenses based on detecting Sybil identities: when a weak identity has limited or no activity history (e.g., interactions with other identities or information they post), the defenses lack sufficient information to determine whether the identity is a Sybil or an inactive non-Sybil. This limitation allows adaptive attackers to create and stockpile large num-

ber of Sybil identities with limited prior activity and use them for tampering crowd computations. If the tampered computations involve legitimate content (e.g., promoting a real business on Yelp for a fee, as opposed to promoting malware links on Twitter), it can be hard to detect the tampering (because the act of recommending a real business is by itself not a sign of Sybil activity).

Given the basic limitation of existing defenses, in this paper, we propose to address Sybil attacks on crowd computations, by moving from *detecting individual Sybil identities* to directly *detecting crowd computations that have significant levels of Sybil identity participation*. Our approach, Stamper, is based on a realization that even when it is fundamentally hard to differentiate between individual Sybil and non-Sybil identities, large *groups* of Sybil and non-Sybil identities can be differentiated. Our approach is based on two key insights:
Key insight 1: If an attacker tampers a computation using a large number of Sybil identities with limited activity, it would result in a *distributional anomaly* or a statistically significant deviation in the distribution of the activity-levels (e.g., number of reviews posted or number of friends formed) of the identities participating in the computation. By analyzing the statistical distributions of the activity-levels of all the identities participating in a crowd computation, we can easily detect such tampering. While there is prior work on detecting malicious activity in crowd computations on e-commerce sites [19,40] and peer-to-peer search networks [28] that looks for anomalies or specific abnormal patterns in feature distributions (where a feature can be some attribute associated with the user activity), our work stands out by providing improved resilience against adaptive attackers.
Key insight 2: To evade detection by the above insight, a determined attacker would have to forge the activities of the Sybil identities under her control to match the distribution of the activity-levels of non-Sybil identities. To be robust against such adaptive attackers, we employ a novel method: we leverage the key observation that even as the attackers forge the activities of their identities, they *cannot forge the timestamps* of their activities (e.g., join date or friend link creation time). The timestamp information is typically recorded by operators for all activities of all identities in the system. By analyzing the statistical distributions of the times when the activity-levels of identities have changed, we can significantly raise the bar for evading detection by adaptive attackers (see Section 3.3).

Another distinguishing feature of Stamper's tamper detection is that it is agnostic to specific attacker strategies: Unlike existing Sybil detection approaches, Stamper does not make specific assumptions about attacker behaviors; instead it uses anomaly detection to detect tampering of any kind. As a result, Stamper can detect computations manipulated by a variety of different attacker strategies, and site operators can choose to further investigate the identities participating in computations flagged by Stamper to detect new and yet undiscovered Sybil identities and attack strategies.

We demonstrate the utility and practicality of Stamper approach by evaluating it over data gathered from two widely-used crowd computing systems: Yelp and Twitter. In the case of Yelp, we evaluate accuracy of Stamper in detecting known tampered computations (already identified by Yelp). We demonstrate that Stamper can detect businesses with highly tampered reviews, independently of the strategies attackers used to manipulate the reviews. Using the Twitter dataset, we demonstrate how a site operator can apply Stamper to detect thousands of previously unknown tampered computations in which Sybil identities were used to boost user popularity. Finally, in section 6 we present a publicly accessible service designed using Stamper to detect tweet content in Twitter with tampered popularity.

2. RELATED WORK AND MOTIVATION

Weak identities and Sybil attacks The systems that Stamper targets allow users to create identities, and require all interactions to be conducted via these identities. Many systems do not require that identities be certified by a trusted authority (to lower the sign-on overheads), and instead only require identities to be created with few credentials (typically, an email address and a solved CAPTCHA). Such identities, known as *weak identities*, are the vector for Sybil attacks, as the small amount of work required to create an additional identity makes it possible for an attacker to create many Sybils. Recent studies show that there are thriving underground "blackmarket" services, where human users or bots can be "hired" to create fake identities [27,39]. These Sybil identities are then profitably used to manipulate crowd-sourced information, such as followers in Twitter [4, 33], reviews in Yelp [3], or content likes in Facebook [2,35].

Limitations with detecting individual Sybil identities The traditional approach to detect if a computation is manipulated involves determining *which* of the participating identities are Sybils. Identities detected as Sybils are then suspended and their contributions to computations are nullified.

Significant recent research has focused on identifying Sybil identities in the system. A large body of work applied machine learning techniques to distinguish between behaviors (activities and profile characteristics) of Sybil and non-Sybil identities [12,26,37,38]. While most approaches in this space use supervised machine learning schemes, work by Wang et al. [37] and Viswanath et al. [35] proposed unsupervised learning schemes to detect Sybil identities. Another body of work has focused on detecting individual Sybil identities by leveraging the structure of the social network graphs formed by Sybils and non-Sybil connecting to one another [36].

However, all these approaches to detect Sybil identities suffer from a fundamental limitation: because weak identities are not backed by some external trusted authority, at their core, all Sybil detection schemes have to rely on analyzing an identity's activities (e.g., interactions with other identities or information they post) to determine if an identity is Sybil. As a result, if an identity has limited or no activity, the schemes lack sufficient evidence to determine if the identity is Sybil or non-Sybil. Studies have shown that many honest users create identities in online systems, but rarely use them [5].

Attackers can take advantage of the above limitations of Sybil detection schemes to create hard-to-detect Sybil identities with only legitimate or limited past activity. An attacker could stockpile a large number of accounts over a period of time, which can later be used to launch hard-to-detect attacks on computations.

Strength in numbers Given the inherent difficulty in determining whether an *individual identity* is Sybil, we pro-

pose to shift the focus to detecting whether a *group of identities* participating in a computation are likely to have Sybil participants.

Few works have explored techniques for preventing Sybil-tampering of computations directly. Prominent among them are DSybil [42], SumUp [34], and Iolaus [23], which work by preventing or discounting votes based on trusted *guides* (DSybil) or the social network (SumUp and Iolaus). Unfortunately, these systems rely on assumptions that do not always hold in a generic crowd-sourcing system. For example, many crowd-sourcing systems do not have social network links interconnecting identities (as assumed by SumUp and Iolaus) and in many systems, a majority of users do not rate many items (preventing the assignment of guides in DSybil).

Prior work has also examined detecting product rating manipulation in e-commerce sites [19, 40] and manipulation of authority scores in peer-to-peer Web search networks [28]. Among them, the most related piece of work is by Feng et al. [19] which explores detecting product rating manipulation in online market places by comparing the distribution of product rating scores of an item to known-good distributions. However, unlike Stamper, the approach by Feng et al. is vulnerable against adaptive attackers as it only considers distribution of product rating scores which can be easily forged to evade detection.

Two other works, SynchoTrap [16] and CopyCatch [13] also focused on analyzing behavior of a group of malicious identities. However, they have a similar limitation where they focus on detecting a specific attack behavior: *loosely synchronized actions*, where a group of malicious identities behave similarly at around the same time. For example, a group of Sybil identities liking a set of Facebook pages at around the same time can be potentially detected by such schemes. In contrast, Stamper does not make any assumptions about specific attacker strategies and thus, has the potential to detect computations tampered using diverse strategies.

3. Stamper: KEY INSIGHTS

3.1 System model and goal

We consider a crowd computing system (e.g., Twitter, Yelp, Facebook) that uses weak identities for its users. A crowd computation can be voting on a given business by a set of identities in Yelp, or promoting a tweet or following a certain identity by a set of identities in Twitter. A site operator is interested in defending against Sybil attacks on crowd computations within the system.

Goal For each computation, the goal of the system operator is to determine whether the set of identities participating in the computation included a *sizeable* fraction of Sybil identities. Stamper design focuses on the core challenge of *robustly detecting tampered computations*. The site-specific actions operators might take against the tampered computations are *not* integral to Stamper design. An operator might choose to suspend (remove) the computations detected as tampered or display the computations at the bottom in site-search results or attach warning labels to them.

Reputation scores We assume that each identity in the system is associated with one or more *reputation scores* that are computed by the operators based on the identity's past activity. Reputation scores can take a variety of forms, and can be computed or obtained by the operators based on "certifications or endorsements", "proofs-of-work", "activity history", or a combination of these. For example, a reputation score could be the number of social network "friends" the identity has in the system, the number of messages it has posted, or the number of endorsements it received from its friends for its work. Given that weak identities are not backed by external trust, reputation scores reflect the system operators' estimation of trust they would place in the identities in the system, i.e., it is less likely (probable) that identities with higher reputation would misbehave (or be Sybils). Note that by definition all newly created identities (Sybils or non-Sybils) will have zero reputation as they have no prior activity.

Threat model We assume that an attacker can create arbitrary number of fake identities in the existing system. However, the attacker does not have unbounded economic resources to create and sustain Sybil identities on every newly created site on the Internet. We allow reputation scores to be *forged*, i.e., attackers may manipulate the different reputation scores of malicious identities they control with different amounts of effort. However, we assume that the site operator keeps detailed historical records of the reputation scores of identities over time.[1] The attacker can also obtain the complete historical records of identities' reputation scores; however, the attacker cannot go back in time and tamper with those records.

3.2 Detecting tampered computations

We describe how Stamper detects tampered computations in two steps below. We first tackle simple attackers and then consider stronger adaptive attackers.

Step 1 In practice, the distributions of the reputation scores of Sybil and non-Sybil identities tend to be quite different. In other words, some attackers today do not expend significant effort to make their Sybil identities similar to non-Sybil identities. Sybil identities as a group, particularly those with limited or no activity, tend to skew towards low reputation scores (as reputation scores are computed based on the identities' activities on the site), while the non-Sybil identities would naturally span a full spectrum of low to high reputation scores.

Insight 1 Due to the above observation, the participation of Sybil identities in a computation tends to distort the reputation score distributions of the nodes participating in the computation. Figure 1 illustrates this insight. It shows the reputation scores for untampered and tampered computations in the Twitter network.[2] The participation of Sybil identities tends to skew the reputation scores towards lower values and has the overall effect of decreasing the entropy in the distribution of scores. It is this difference in reputation score distributions that Stamper leverages to detect Sybil tampering.

We stress that the above insight allows us to determine that a computation has been tampered with even when we cannot determine which of the identities are Sybil. In Fig-

[1]Many site operators today including Facebook and Twitter are known to keep detailed historical records of identities.

[2]These are samples of real untampered and tampered computations in Twitter flagged by Stamper (See Section 5.2).

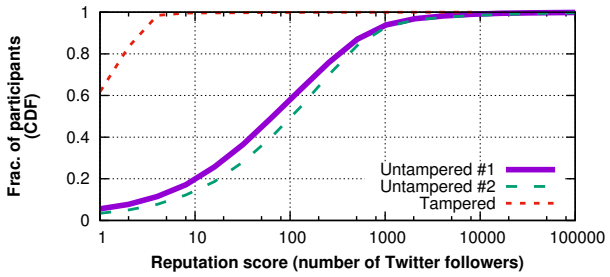

Figure 1: Reputation score (based on number of followers) distribution of tampered vs untampered computations. Most participants in the tampered computation have a low reputation score.

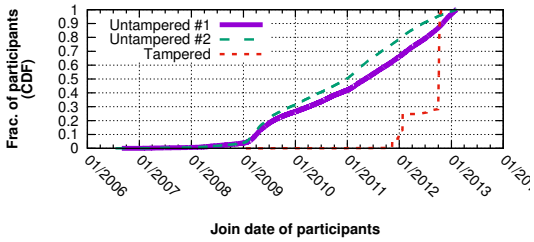

Figure 2: Join date distribution of participants of tampered and untampered computation.

ure 1, even as we infer that the skew towards lower reputation scores is due to Sybils, we cannot tell which of the identities with low reputation scores are Sybils as there *do* exist non-Sybil identities with such low reputation scores as well.

However, this insight alone would not allow us to design an approach that is robust against an adaptive attacker. For example, a determined attacker could expend additional effort to manipulate the reputation scores of her Sybil identities to match the distribution of reputation scores of non-Sybil identities.

Step 2 Even when an attacker can forge a malicious identity's reputation, she can only forge the *present and future* reputation scores of the identity, but she cannot go back in time and forge the past history (i.e., temporal evolution) of the identity's reputation as recorded by the operator. Thus, the distribution of temporal evolution of forged reputation scores of Sybils tend to exhibit distributions that are quite different than non-Sybils'.

Insight 2 To detect potential forging of reputations of Sybil identities by an adaptive attacker, we analyze the temporal evolution of reputation scores of the identities participating in the computation. Specifically, we examine the distributions of times (i.e., dates) when the identities have achieved a certain percentile (e.g., 0%, 10%, 25%, 50%) of their current reputation score.

Figure 2 illustrates this insight. It shows the times when the identities participating in untampered and tampered computations began to acquire reputation in the system (i.e., their join date).[2] The Sybil identities have acquired most of their reputation within the short period of time close to their participation date in the computation, while

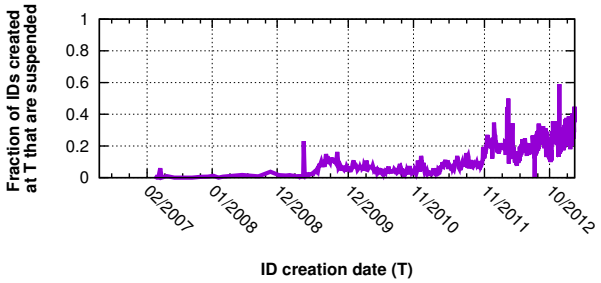

Figure 3: Growth in the fraction of identities in Twitter that are eventually suspended.

the non-Sybil identities have acquired their reputations over a much longer period of time. In fact, a significant fraction of untampered computations have identities with reputation histories dating back to the inception of the Twitter site (in 2006). It is this difference in the distributions of temporal evolution of reputation scores of identities that Stamper leverages to detect Sybil tampering of a computation.

3.3 Robustness and Limitations

In this section, we discuss how Stamper raises the bar for evasion by adaptive attackers and also point out limitations against adaptive attackers.

Robustness 1: Can an adaptive attacker create Sybil identities whose reputation histories match those of non-Sybils? To tamper a computation without being caught by Stamper, an attacker would have to create new Sybil identities and groom their reputations from the inception of the system, i.e., when non-Sybils started being created.

In existing systems like Facebook, Twitter, or Yelp, the attacker is already too late to go back in time and forge identities whose reputation histories date back to the time when these sites came into existence. Figure 3 shows the growth in suspended identities in Twitter, since the time of its inception.[3] In the first two years, Twitter witnessed very few (0.036%) malicious identities. However, once Twitter reached a critical mass of users, it started attracting more attackers and the percentage of malicious identities sharply rises to as high as 40% of all identities created on a single day. So, it is hard for any attacker to gain control of Sybil identities with reputation histories dating back to the early years of these existing systems.

An attacker still has an opportunity to create and groom Sybil identities on newly (or yet to be) created online sites. However, the attacker cannot accurately predict which of the several new online sites are likely to succeed and acquire critical mass of users in the long run. So the attacker would have to create Sybil identities on *all* online sites from their inception to be prepared to launch an attack on any single site in the future. Considering the large number of new online sites that are created everyday, we argue that such attacks are not economically viable in practice.

Robustness 2: Can an adaptive attacker create "sleeper cells" to launch an attack in the distant fu-

[3]We crawled 2.3M Twitter identities that joined Twitter at different points of time since its inception. Twitter API allows us to figure out which identities have been suspended.

ture? A determined attacker can start creating "sleeper" Sybil identities on popular sites like Facebook and Twitter today, with the goal of launching an attack several years down the road. While such attacks are not impossible, we argue that Stamper significantly raises the costs for the attacker making it economically non-viable. Since Stamper checks for temporal evolution of reputation scores, an attacker would need to actively groom the "sleeper" identities to evolve their reputation scores similar to how non-Sybils reputation scores evolve. As different reputation scores of user identities in different systems evolve in complex, unpredictable ways, an attacker would have to constantly track and mimic these changes. The difficulty of grooming such identities would be reflected in their cost for the attack.

To test our hypothesis about attacker costs for creating identities with high reputation histories, we collected pricing information for Sybil identities in Twitter and Facebook, by manually inspecting postings in 8 online black-market services (that we found via google search), where such identities are sold. A Facebook (Twitter) identity with no reputation costs \$0.51 (\$0.09), while those with 4 years of age cost \$15 (\$1) and those with 5000 (200) real and active friends (followers) costs \$150 (\$5). While the data we gathered does not constitute a rigorous proof, it indicates that identities with long running and high reputations could cost 10 to 100 times or more than newly created Sybil identities with no reputation.

Robustness 3: Can Stamper fundamentally alter the arms race between Sybil attackers and defenses? The root cause of arms race between Sybil attackers and defenses today is that every time a Sybil identity is detected and suspended, attackers can not only create a new Sybil identity and regain their lost attack power, but they can also derive knowledge about how to evade detection. With Stamper, every malicious identity suspended by the site operator would represent a loss in the power of the attacker because the attacker cannot replace the suspended identities with newly created identities. If used in an attack, Stamper would be able to detect the differences in how reputations of identities evolved over time for older and newer identities.

Most site operators today proactively deploy "spam filters" that over time detect Sybil identities and suspend them. While these spam filters are far from detecting all Sybil identities in a timely manner, Stamper fundamentally shifts the balance of the arms race in favor of these defenses because every suspended identity results in a near permanent reduction in attack power, which can be regained with a new Sybil identity only after waiting for the entire generation of existing identities to leave the system (see Robustness 2 above).

Limitation 1: Can Stamper detect attacks using non-Sybil identities that are incentivised to collude or whose login credentials are compromised? Stamper cannot provide the robustness guarantees discussed above for attacks involving non-Sybil identities that are compromised by an attacker or that have an incentive to collude with one another (e.g., to boost each other's popularity). However, there is still hope; Stamper would still be able to detect tampering as long as the colluding or compromised identities are not carefully chosen in such a way that the distribution of reputation scores and their temporal evolution match that of non-Sybil identities. In practice, it may not be easy for an attacker to selectively target and compromise non-Sybil identities with varied levels of reputation scores. In fact, in our evaluation Section 5.2, we show that Stamper is able to detect identities colluding to follow one another in Twitter, because their collusion distorts the distribution of their reputation scores, which is easily flagged by Stamper.

Limitation 2: Can Stamper detect computations that involve only a few identities or that have been tampered using only a few Sybils? At its core, Stamper relies on identifying *statistically significant* deviations in distributions of reputation histories of identities participating in a crowd computation. So the robustness guarantees of Stamper do not hold when the number of participating identities is too small or when the degree of tampering is small. In practice, we show that Stamper can be used to detect tampering of computations with 100 or more participants (see Section 5.1). We also show that Stamper is very robust in detecting highly tampered computations (e.g., $> 50\%$ of identities are Sybil), but when the computations are tampered only to a small extent (e.g., $< 10\%$ of identities are Sybil), the detection accuracy suffers. While this is a fundamental limitation of Stamper's approach, it is worth noting that in practice, system operators would be more concerned about detecting heavily tampered computations than lightly tampered ones.

4. Stamper DESIGN

We design Stamper to satisfy the following requirements for a practical design: (i) *robustness:* any computation flagged as being tampered with should have been tampered with very high probability and any tampered computation has a good chance of being detected; (ii) *generality:* the system should be able to detect Sybil tampered computations independently of the attack method used.

Notation Let sets A, M, and H respectively represent all identities, all Sybil identities, and all non-Sybil (honest) identities in the crowd computing system (e.g., Twitter, Yelp, or Facebook) such that $\mathsf{A} = \mathsf{H} \cup \mathsf{M}$. We assume that each identity is associated with a set of reputation scores $\mathsf{R} = \{R_1, R_2, \ldots, R_\ell\}$, which are computed by the operators based on the identity's activity in the system to date. For a given set of identities that participated in a crowd computation c, we denote by $R_i(c)$ to be the probability distribution (or density) function (PDF) of the values of the reputation score R_i of the identities in computation c.

The system operator is interested in defending against Sybil attacks on a set of crowd computations $\mathsf{C} = \{c_1, c_2, \ldots, c_n\}$ Let sets C_i, $\mathsf{M}(C_i)$, $\mathsf{H}(C_i)$ respectively represent all identities, Sybil identities, and non-Sybil identities that are involved in computation c_i.

4.1 Design overview

Our goal is to design a "detector" that can check if a given large crowd computation c_i was tampered with by Sybil identities, i.e., whether the (unknown) Sybil identities $\mathsf{M}(C_i)$ constitute a significant fraction of all identities C_i participating in the computation c_i.

We compute the *relative entropy* or divergence between two distributions using a statistical measure called the Kullback—Leibler (KL) divergence [24]. The choice of KL-divergence as the statistical measure is not fundamental to the application of Stamper. We could have used other sta-

tistical distance measures [1], but as we show in our evaluation, KL-divergence is quite sufficient for our purposes. KL-divergence ranges from 0 (identical distributions) to ∞ (highly differing distributions); the (symmetric) divergence between two distributions P and Q is denoted by $\mathsf{KLD}(\mathsf{P},\mathsf{Q})$, where

$$\mathsf{KLD}(\mathsf{P},\mathsf{Q}) = \sum_{i=1}^{r}\left(\log(\frac{\mathsf{P}(i)}{\mathsf{Q}(i)})\mathsf{P}(i) + \log(\frac{\mathsf{Q}(i)}{\mathsf{P}(i)})\mathsf{Q}(i)\right)$$

Our insight suggests that in practice, the KL-divergence between distributions of untampered computations would be low, while those between untampered and tampered computations would be anomalously high.

4.1.1 Detecting anomalous distributions

We use *anomaly detection* [22, 25, 35] techniques to separate out the "outlier" or "anomalous" distributions of reputations scores (and their temporal evolution) observed for tampered computations. Specifically, we apply a variant of anomaly detection known as *semi-supervised* anomaly detection [30], where the site operator has *a priori* knowledge of a small subset of crowd computations, $\mathsf{UC} = \{uc_1, uc_2, \ldots, uc_k\}$ that are largely untampered with by Sybil identities.

We first analyze the KL-divergence in the distribution of a reputation score R_j between the known untampered computations. If we find that the distributions of most untampered computations lie within some small threshold divergence T_j from one another, then we could declare any other computation whose distribution lies far outside the threshold T_j as potentially tampered.

When identity participation is unbiased We can offer strong theoretical guarantees on the choice of the threshold T_j, if participants in any untampered crowd computation c_i are drawn uniformly at random (without any bias) from the set of all non-Sybil identities H in the system. Under the unbiased participation assumption, the probability distributions of the reputation scores of identities participating in all large untampered computations are guaranteed to converge to the same distribution. Specifically, as the size of a computation c_i grows, the distribution of reputation scores $R_j(c_i)$ quickly approximates the distribution of reputation scores for all non-Sybil identities $R_j(\mathsf{H})$. Formally, for all c_i such that $C_i = \mathsf{H}(C_i)$, and for some small ϵ,

$$\exists\ s \quad \text{s.t.} \quad \forall i, |C_i| > s\ \cap\ \ \mathsf{KLD}(R_j(\mathsf{H}), R_j(C_i)) < \epsilon.$$

We refer to s as the *size threshold* for the crowd computations. In fact, Roy [29] studied the thresholds theoretically as well as empirically and proved an upper bound of $1/|C_i|$ on KLD for a sampled distribution of size $|C_i|$. Thus, if an untampered computation involves over 100 or 1,000 identities, the KLD between the reputation score distributions of the computation participants and the non-Sybil identities will be lower than 0.01 or 0.001, respectively. As a result, a simple strategy for detecting whether a given large computation c_i (i.e., $|C_i| > s$) has been tampered with is as follows: First, select some *a priori* known untampered computation c_u of size greater than s. Then, compute the divergence in the distributions of reputation score R_j between the given computation and known untampered computation, i.e., compute $\mathsf{KLD}(R_j(c_u), R_j(c_i))$. If the divergence is greater than the divergence threshold $1/s$, declare the computation c_i as tampered (with high probability).

When identity participation is biased In practice, many crowd computations draw a biased population of identities: For example, in Yelp, many reviewers of businesses in San Francisco are likely to be drawn from San Francisco. Without the unbiased participation assumption, we cannot offer any *theoretical* guarantees on convergence of distributions of untampered computations. However, in practice we often observe that the *distributions for untampered computations are far closer to one another than they are to tampered computations*. We validate this claim using real-world data from Yelp and Twitter in the evaluation sections 5.1 and 5.2.

In the case of biased participation, we first derive a *reference* or expected distribution by "averaging" the distributions of known untampered computations and then select a KL-divergence threshold T_j that encompasses most, if not all, the untampered computations. To detect whether a given large computation c_i is tampered with, we compute its KL-divergence from the reference distribution. If it is larger than the threshold divergence T_j, we declare the computation c_i as tampered (with high probability).

While we defer the precise details of the threshold selection to Section 4.2, we make two observations on the choice of the divergence threshold. First, if for some reputation score R_j, the distributions of untampered computations do not converge in practice, then the observed threshold divergence T_j between the untampered computations would also naturally be quite large, and consequently there would be little risk of an untampered computation flagged as tampered. Thus, the risk of untampered computations being flagged as tampered is low, even when the distributions of untampered computations do not converge. Second, by raising and lowering the threshold T_j, an operator can trade-off between the precision and recall in detecting tampered computations. Depending on the application scenario, operators can either choose a more or less conservative threshold.

4.2 Detailed Design

The operator would deploy Stamper as follows:

1. Creating a pool of reputation scores The first step in deploying Stamper involves choosing a set of reputation scores $\{R_1, R_2, ..., R_l\}$ that can be computed for each identity in the system. Identities start with low (zero) reputation scores when they are created and can earn higher reputations over time. We do *not* assume that reputation scores are unforgeable: different reputation scores of an identity may be manipulated by the attacker with different amounts of effort.

2. Building a reference (expected) distribution To build a reference distribution for a given reputation score R_j, we first compute the distribution of the reputation score for each known-untampered computation (i.e., calculates $R_j(uc_i)$ for each $uc_i \in \mathsf{UC}$). Now, these distributions are aggregated into a single reference distribution $\mathsf{R}_j(\mathsf{UC})$ using a *linear opinion pool* [17] model. We do so using a fair weighting scheme such that each crowd computation contributes a fair share towards building the final reference distribution. Formally, the reference distribution $\mathsf{R}_j(\mathsf{UC})$ is defined as $Pr[v \leftarrow \mathsf{R}_j(\mathsf{UC})] = \frac{1}{k}\sum_{i=1}^{k} Pr[v \leftarrow \mathsf{R}_j(uc_i)]$

3. Selecting a threshold We now compute the KL-divergence of each of the crowd computations in set C from that of the reference distribution. We will obtain a range

of KL-divergence values and will select a threshold T_j, such that KL-divergence values greater than T_j is anomalous with respect to the rest of KL-divergence values. To select this threshold, we use a simple statistical technique called the *box plot rule* [14] defined as follows: Let $Q1$ and $Q3$ be the lower and upper quartile respectively, for the KL-divergence values. A KL-divergence value is an outlier if it lies beyond the *upper outer fence:* $Q3 + 3 * (Q3 - Q1)$. We select the upper outer fence of the distribution as the threshold T_j.

4. Detecting anomalous computations With T_j and $\mathsf{R}_j(\mathsf{UC})$, it is straightforward to detect anomalous computations. For a given computation c_i, the operator simply calculates the KL-divergence between $R_j(c_i)$ and $\mathsf{R}_j(\mathsf{UC})$; if it is higher than T_j, the computation is flagged as anomalous. In fact, the higher divergence (above the threshold), the more anomalous the computation turns out when compared to the rest of the computations. The operator can experiment with the tradeoff of catching more tampered computations (when using a lower KL-divergence threshold) versus improving the efficiency of workers (when using a high KL-divergence threshold).

Operators typically use human workers to examine suspicious accounts or activities once they are flagged by their defense mechanisms [15]. Stamper can guide operators to focus the attention of their human verifiers on a set of flagged computations to verify if they are tampered with. More importantly, while Stamper has been designed to detect computation tampering, it can be used in practice for a broader range of Sybil defense tasks. The operator can manually investigate the anomalous computations—as they have a higher chance of containing Sybils—to further discover new Sybils and previously unknown attacker strategies. We demonstrate this in Section 5.2.2 where we investigate the identities that participate in tampered computations. However, it should be noted that an investigation phase is common in deployed defense schemes and it is not part of the core Stamper deployment workflow.

5. Stamper EVALUATION

This section presents two case studies of applying Stamper in two popular systems, namely Yelp and Twitter.

5.1 Case 1: Yelp review tampering

Goal: Find businesses with tampered reviews Yelp is a popular local directory service, where users can search for businesses in a given locality and retrieve crowd-sourced reviews and ratings for those businesses from other users. As Yelp is becoming popular, businesses (e.g., restaurants) have an incentive to manipulate their reviews and ratings in their favor. Today, there are plenty of black-market services [10], where one can easily buy Yelp reviews for a cheap price (e.g., three reviews cost $74.85 in one such service). The crowd computation that we are interested in is the set of identities that rate a given business in Yelp. Our goal is to evaluate how effectively Stamper can be leveraged to detect attacks that tamper the computation, i.e., detect businesses that have manipulated reviews.

For evaluating effectiveness of Stamper we leverage Yelp's *review filter* [8, 9] feature to obtain "ground truth" for tampered reviews. Yelp filters suspicious reviews to defend against fake reviews. It should be noted that as is the case with many online defense schemes deployed today, Yelp acknowledges that their review system is not perfect and may not be able to detect all types of tampered reviews and may even sometimes wrongly flag legitimate reviews. However, for the purpose of this analysis, we will consider a business to have tampered reviews if Yelp filters at least one review of the business. Note that we do not have any knowledge about specific strategies used by attackers of the filtered reviews (i.e., did the attacker create multiple fake accounts to tamper reviews or did she incentivize real users to write fake reviews in return for a monetary reward).

More precisely, we investigate the following three questions: (i) How easy or difficult is it to apply Stamper to detect review tampering in Yelp? (ii) Does the key requirement that distributions of reputation scores of large untampered computations converge (while those of large tampered computations diverge) hold in practice in Yelp? (iii) Can Stamper detect most of the *highly tampered* computations (businesses with a majority of reviews filtered) at a low false positive rate (fraction of businesses with no filtered reviews flagged)? Recall that Stamper is designed to detect *highly tampered* computations and cannot guarantee detection of computations tampered only to a small extent (see Section 3.3).

Data gathered We used Yelp data gathered by Kakhki et al. [23] in May 2012, which we updated with our own data gathering crawl in March 2013. This dataset consists of all businesses on Yelp in the city of San Francisco at that time. This includes 30,339 businesses with a total of 1,655,385 ratings from 340,671 reviewer identities. Each rating consists of a score from 1 to 5 stars. These ratings also include those *filtered* by Yelp's automated review filter. In total, Yelp filtered 195,825 (or 11.83%) ratings. As Stamper has been designed to infer tampering in large computations involving more than a certain number of identities, we threshold the size of the computation at 100 for Yelp. There are 3,579 businesses with more than 100 reviews. Out of these 3,579 businesses, there are 54 businesses which did not have a single review filtered by Yelp. We consider these 54 cases as untampered computations. Also, for each reviewer we collected information about a variety of reputation scores. (See the first column of Table 1.)

5.1.1 *Ease of deploying* Stamper

The four steps that constitute Stamper detection strategy (outlined in Section 4.2) can be applied in a straight-forward manner with very little overhead.

1. Creating a pool of reputation scores The first column in Table 1 lists all the 8 reputation scores used in our evaluation; e.g., the reputation score in the 8th row is a measure of the number of times reviews by an identity are marked useful by other identities in the service. To tamper a crowd computation by forging this reputation score, an attacker would have to put additional effort to boost the reputation for the malicious identities employed in the attack by obtaining a certain number of endorsements (by getting reviews marked useful) from other identities.

2. Building reference distributions We select businesses which had no (zero) review filtered as the set of untampered computations. There are 54 such businesses (with zero reviews filtered). Even though the number of untam-

Reputation score	# flagged	Percentage of computations flagged				
		0% filtered	(0,10]% filtered	(10,30]% filtered	(30,50]% filtered	> 50% filtered
# photos	158	**5.6 (3/54)**	0.2 (5/2280)	6.6 (72/1089)	32.9 (27/82)	**68.9 (51/74)**
# first badges	141	**0.0 (0/54)**	1.3 (30/2280)	4.4 (48/1089)	20.7 (17/82)	**62.2 (46/74)**
# fans	139	**0.0 (0/54)**	0.6 (14/2280)	5.0 (54/1089)	28.0 (23/82)	**64.9 (48/74)**
# compliments	173	**1.9 (1/54)**	0.7 (15/2280)	6.2 (67/1089)	35.4 (29/82)	**82.4 (61/74)**
# reviews marked funny	157	**0.0 (0/54)**	0.7 (15/2280)	5.0 (54/1089)	28.0 (23/82)	**87.8 (65/74)**
# reviews marked cool	174	**0.0 (0/54)**	1.0 (22/2280)	5.6 (61/1089)	35.4 (29/82)	**83.8 (62/74)**
# friends	227	**0.0 (0/54)**	0.2 (4/2280)	9.7 (106/1089)	56.1 (46/82)	**95.9 (71/74)**
# reviews marked useful	224	**3.7 (2/54)**	0.5 (11/2280)	9.2 (100/1089)	51.2 (42/82)	**93.2 (69/74)**
All scores combined	362	**5.6 (3/54)**	3.0 (68/2280)	14.8 (161/1089)	70.7 (58/82)	**97.3 (72/74)**

Table 1: Computations with varied levels of filtered reviews flagged by Stamper. Stamper flags most of the *highly tampered* (>50% filtered) computations while flagging very few (3/54) untampered computations.

pered computations might seem small, they have a large number of reviewers (14,223 reviewers) who wrote reviews for them.

3. Selecting a threshold We compute KL-divergence values for all 3,579 businesses from the reference distribution for each reputation score. Then, for each reputation score, using the box plot rule, we estimate a KL-divergence threshold for flagging anomalies; e.g., in the case of the reputation score, *#times review is marked useful* (we will call this as the number of review endorsements), we estimate a threshold of 1.2.

4. Detecting anomalous computations If the KL-divergence computed for a business is greater than the divergence threshold for any reputation score, the computation is marked as anomalous. The second column of Table 1 shows the number of businesses whose reviews have been flagged as tampered by Stamper.

The above discussion demonstrates how easily Stamper can be applied by operators of crowd computing systems today.

5.1.2 *Detectability of tampered computations*

We now investigate whether a key assumption behind Stamper design holds in practice. Specifically, we verify if the distributions of reputation scores of large untampered computations in Yelp tend to converge, while those of tampered computations diverge. Figure 4 shows the distribution of KL-divergence values using the endorsement count reputation score for untampered computations and computations with different levels of filtered reviews. Note that, computations with zero and with less than 10% reviews filtered show low KL-divergence values from the reference distribution, indicating a good convergence in the reputation score distributions. In fact, for 90% of untampered computations their divergence values are less than or equal to 0.36. While for tampered computations (computations with more than 20% and 50% reviews filtered), the KL-divergence values are higher and shows a diverging trend. We observe a similar trend for other reputation scores listed in Table 1.

5.1.3 *Robustness of* Stamper *tamper detection*

Next we investigate the robustness of Stamper detection. For the rest of the analysis, we divide businesses into five categories based on the level of filtering: 0% filtered (or untampered), 0 to 10%, 10 to 30%, 30 to 50%, and more than 50% filtered. We consider computations with more than 50% reviews filtered to be highly tampered. Out of a total of 3,525 businesses with at least one filtered review, there are 74 businesses that are highly tampered.

1. Stamper can detect most of the highly tampered computations The last column in Table 1 shows the fraction of highly tampered computations that are flagged. By combining all 8 reputation scores (a computation is flagged if it is flagged by at least one reputation score), we detect more than 97% of highly tampered computations. It is interesting to note that combining multiple reputation scores can help to catch more tampered computations. Stamper also manages to catch a significantly high fraction (over 70%) of computations with 30 to 50% reviews filtered.

2. Stamper has low false positives The third column in Table 1 shows the fraction of untampered computations flagged. By combining all 8 reputation scores, we observe a false positive rate of only 5.6%. While interpreting this false positive rate, it is important to keep in mind that Yelp's review filter is not perfect and could have potentially missed flagging some fake reviews.

5.1.4 *Discussion*

Why is Stamper useful for a system like Yelp? Note that Stamper does not detect individual suspicious reviews. While this might sound like a limitation, Stamper can still be useful for Yelp in flagging businesses with highly tampered reviews. For example, Yelp is known to suspend businesses that were caught buying reviews [11], and display a warning when a user visits a business page suspected of tampering reviews [7]. Using Stamper, Yelp can do so even without detecting individual suspicious reviews as they might be very hard to detect for various reasons. For example, Yelp went

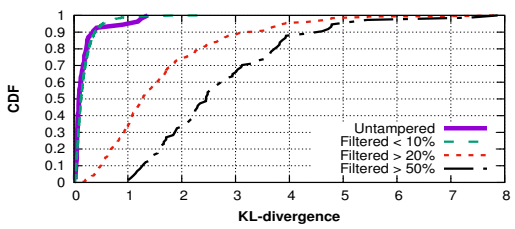

Figure 4: Distribution of KL-divergence values for untampered and tampered computations using number of review endorsements in Yelp.

to the extent of conducting sting operations to catch businesses trying to buy fake reviews [6] because we suspect that such type of tampering is very hard to detect by analyzing reviewer behavior or the content of their reviews. With Stamper, Yelp has the potential to catch highly tampered computations even with very minimum or no information about the behavior of the reviewers. Another huge advantage of our scheme is that compared to prior machine learning approaches, Stamper can detect highly tampered computations in Yelp without training on any pre-identified fake reviews.

Leveraging temporal evolution of reputation scores We tried to find anomalous computations by analyzing the temporal evolution of reputation scores. We used the timestamp at 0th percentile reputation, which is the join date of the user. Stamper flagged only 2 highly tampered computations (already caught by the other reputation scores) using join dates. We suspect that attackers on Yelp are not trying hard to forge their reputation scores today, and we are able to detect most of the highly tampered computations using simple reputation scores. In the next case study, we observe that temporal evolution of reputation scores are very helpful in catching adaptive attackers.

5.2 Case 2: Twitter follower tampering

Goal: Find Twitter users with fake followers In Twitter, to obtain real time information posted by specific users, users typically *follow* those users. Today, the influence of a user is often estimated by counting the number of followers. As a result, there are strong incentives for users to acquire more users to follow them and there have been numerous reports of follower count manipulations [31]. Thus, the crowd computation that we are interested in is the set of identities in Twitter that follow a given Twitter identity. Our goal is detect attacks that manipulate the computation, i.e., detect identities that have tampered follower counts.

We use this case study to showcase Stamper's capability of detecting yet unknown tampered computations. This provides an opportunity to evaluate how system operators (who in practice would not have a priori ground truth information about tampered computations) might use Stamper. More precisely, we investigate the following three questions: (i) How easy or difficult is it to apply Stamper to detect computation (follower count) tampering in Twitter? (ii) Can system operators analyze the computations flagged by Stamper further (potentially manually) to detect (potentially new) patterns of Sybil attacks? (iii) Can the newly discovered Sybil attack patterns be used to uncover more Sybil identities?

Data gathered We target detecting tampering of follower-counts only for popular Twitter user identities with more than 1,000 followers. We obtained the Twitter-UIDs (unique identifiers) of all users with more than 1,000 followers in all of Twitter (as of July 2012) from a research group which collected this data for a separate study [20]. This dataset contained 2,100,851 identities. We selected a random sample of 70,000 of these identities and gathered profile information of all their followers. Some of these accounts no longer existed on Twitter and their information could not be collected. In total, we discovered (in aggregate) over 176M followers for 69,409 of these users.

5.2.1 *Ease of deploying* Stamper

We briefly discuss the steps that constitute Stamper procedure for this case study.

1. Creating a pool of reputation scores We select *number of followers* as a reputation score to build our reference distribution (i.e., we consider the distribution of the number of followers *of the followers*). To account for the cases where an attacker forges this reputation score, we consider the temporal evolution of the reputation score (i.e., the distribution of times at which the identity acquired 0th, 25th, or 50th percentile of their reputation). Since it was easy for us to gather the timestamps at which the identities started building their reputation (i.e., the date at which the identities "joined" the service—corresponding to the 0th percentile of their reputation—we use the join dates of identities to build the reference distribution.

2. Building reference distributions In Twitter, we assume that the accounts verified by Twitter[4] do not knowingly tamper their follower counts. We use them as the set of known untampered computations. We randomly sample 30,000 verified accounts from the list of Twitter verified accounts with more than 1,000 followers, and crawled profiled information of their 266M followers to derive the reference distribution.

3. Selecting a threshold We compute KL-divergence values for all the 69,409 identities and estimate a KL-divergence threshold of 7.88 and 5.79 using the follower count reputation score, and join date, respectively.

4. Detecting tampered computations Among the 69,409 popular users, using follower counts, Stamper flags 620 users and using join dates, Stamper flags 1,129 users as having tampered follower counts. When we examine the overlap between computations flagged using follower count and join date (i.e., the 0th percentile of the follower count), it is very low, consisting of only 49 computations. Thus, using join dates, Stamper is able to flag 1,080 users (with potentially tampered follower counts) who were not flagged using the follower count reputation score. These 1,080 users were able to successfully hide or evade detection when using the follower count reputation score. This finding further shows the advantages of using unforgeable timestamps to detect tampering. Our discussion above once again demonstrates the ease of deploying Stamper.

5.2.2 *Investigating anomalies to detect new attacks*

For manual investigation, we randomly sample 50 computations out of each group of anomalous computations flagged using follower count and join dates, respectively. Three graduate students with prior experience in investigating suspicious identities in social networks spent roughly 15 to 20 minutes per computation for investigation.

First, we try to understand the characteristics of the distribution for each anomalous sample. Second, we attempt to localize our analysis to a subset of the identities within the tampered computation that are most likely to be Sybils. We can find such a subset by looking for regions within the anomalous distribution where it exhibits maximum diver-

[4]Twitter vouches for the authenticity of a small portion of all identities (43,901 identities as of April 2013) through an offline verification process.

gence from the reference distribution. To identify potential Sybil identities, for each candidate account, we analyze the Twitter profile picture, name, bio, content of tweets posted (including URLs posted), follower and following count, and profiles of followers of the account.

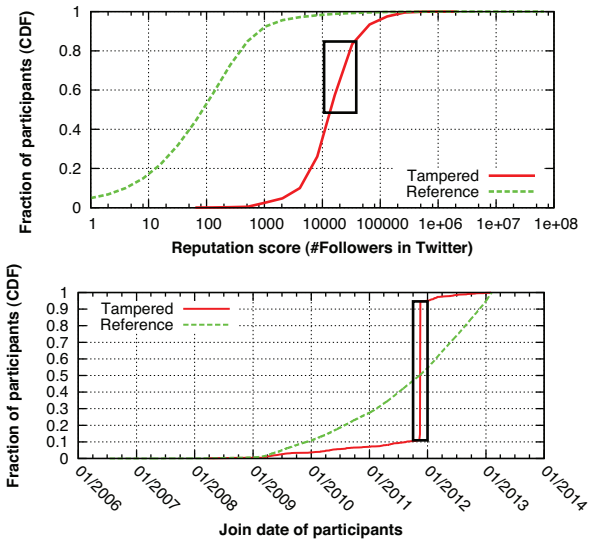

Figure 5: Detecting suspicious participants of a tampered computation. The highlighted region contains the identities suspected of tampering the computation.

Investigating anomalies in follower count distributions On investigating the distribution of follower counts we noticed two distinct patterns. In the first pattern, we found that most of the followers have very low reputation (e.g., almost all had less than 10 followers). 2 out of 50 computations exhibit this pattern. These followers look like fake accounts (fake looking profile, pictures and tweets talking about following activity) and some were already suspended by Twitter. While we would expect tampering to be carried out by unpopular Sybil identities, we were surprised to find only two such instances. The remaining 48/50 computations showed a different distribution pattern–an example is shown in Figure 5 (top figure). When analyzing these computations, we surprisingly discovered that these identities appear to be popular users (i.e., more than 1,000 followers) tampering their follower counts by *colluding* with one another and exchanging links with "you follow me, I follow you" deals. Such activity is referred to as "farming" links on the Twitter network. Link farming [21] is a well-studied problem in Twitter and their defining traits are as follows: they have a large number of followers (more than 1,000), a following per follower ratio in the range [0.9,1.1] and a majority of their followers satisfy these criteria as well. The identities we investigated match these link farmer traits. Using this definition we analyzed all the followers of each of the 48 computations, and found that all computations had at least 32% of their followers that matched the criteria for link farmers. In fact, for 44 out of 48 cases, a majority of their followers are link farmers.

Our analysis above reveals that even though Stamper has been designed to catch computations tampered by Sybil attacks, it is able to detect a broader category of attacks, including *colluding attacks* by non-Sybil identities. However, we do not claim that our robustness guarantees would extend to such non-Sybil attacks.

Investigating anomalies in join date distributions We analyzed the distributions of the 50 random computations in this case and noticed that for a majority (33/50) of the computations, a significant fraction of the participating identities are tightly clustered in the time domain. Figure 5 (bottom figure) gives an example of such a tampered computation. More specifically, we check if at least 10% (12%) of the followers of the identity joined Twitter within a single day (or week).[5] In contrast, the join dates of identities in the reference (untampered) distribution are spread out over many years. The tight-clustering in the time domain suggests that these identities are possibly created by a Sybil attacker on a single day and then pressed into attack soon after.

To further test our hypothesis, we bought followers for 10 different Twitter identities under our control from 10 different online marketplaces. We discovered these services with a simple keyword search (e.g., "buy Twitter followers") on search engines like Google. In each case, we paid to receive 1,000 followers. When we analyzed the timestamp distribution of followers bought from the black-market for the 10 accounts, we observe that for 9 out of 10 accounts, a vast majority of followers were created on the same day or on a handful of days. This observation supports our hypothesis that identities discovered in anomalous distributions of join dates are Sybils from whom links have been bought.

We further analyzed the remaining 17 (out of 50) computations that did not exhibit tight clustering at a day or week granularity. On manual investigation of followers in the most divergent region, we found that 15/17 computations had very suspicious followers and are most likely tampered computations. Many followers look like fake profiles with no profile picture, names with specific patterns, meaningless tweet content, and some even had malware links in their tweets. An interested reader can browse through more details of the manual analysis of these 15 suspicious computations on this page: `http://trulyfollowing.app-ns.mpi-sws.org/local/stamper/tampered_fcounts.html`. Note that our manual investigation provides detailed information about why we think a computation looks suspicious along with a sample of suspicious participants of the computation.

5.2.3 *Detecting new Sybil identities*

We now show how an operator can leverage the newly discovered attacker strategies to detect new Sybil identities. We propose to identify cases of follower tampering by analyzing the join date distributions of followers of a given identity and checking if a non-trivial fraction of their followers joined Twitter within a small window of time (we use the same thresholds discussed in earlier section). We applied this technique to detect potential follower tampering activity in Twitter to over 2.1M identities in Twitter

[5]Note that we choose conservative thresholds where we found that over 95% of 69K computations did not exhibit this level of tight clustering in the time domain. To validate our thresholds of 10% followers in a day (and 12% in a week), we monitored accounts that exhibited tight clustering and those that did not, for 6 months. Identities forming clusters had a high Twitter suspension rate of 36% compared to a low suspension rate of 0.38% for the other accounts.

with more than 1000 followers. We detect 89,728 identities as having tampered follower counts. Interested readers can browse the data about these 89,728 identities at the site: `http://trulyfollowing.app-ns.mpi-sws.org/`. From these flagged computations, we identified over 23 million Sybil followers whose join dates fall within a small window of time.[5]

5.3 Ethics

All the data about user activity collected from Yelp and Twitter are publicly visible information. All money we paid to acquire followers from the black-market were exclusively for Twitter accounts under our control and set up for the sole purpose of conducting the experiments in the paper. Overall, we ensured that no user or page on Yelp and Twitter was abused or benefited as a result of our study.

6. Stamper DEPLOYMENT

Finally, to demonstrate the effectiveness of Stamper in the real world, we deployed a public online service at `http://trulytweeting.app-ns.mpi-sws.org/` which detects *tampered tweet promotions* in Twitter. Today, there are strong incentives for users to artificially boost popularity of their posts by hiring Sybil identities to promote their content. In this service, we are interested in three types of crowd computations involving a set of identities that: (1) tweet about a particular topic (described by a set of keywords) (2) tweet a URL, or (3) retweet a tweet. Our goal is to detect attacks that manipulate such computations, i.e., detect content (topic, URL or tweet) that is promoted by Sybil identities. Our service lists currently *trending topics*[6], popular URLs and tweets that are tampered and also provides a real time search interface to check arbitrary URL or topic computations for tampering in Twitter. We encourage interested readers to test the service to understand the potential and practicality of Stamper.

7. CONCLUSION

In this paper, we tackle the challenging problem of detecting when computations on crowdsourcing systems like Twitter or Yelp have been tampered by fake (Sybil) identities. We have advocated a fundamentally different approach called Stamper that can detect whether a computation has been tampered even when it is not feasible to detect which of the individual identities participating in the computation are Sybil. The key insight that enables our approach is that large statistical samples (groups) of Sybil and non-Sybil identities exhibit very different characteristics. We have leveraged this insight to design Stamper to (i) detect tampered computations and raise the bar for defense against adaptive attackers and (ii) detect computation tampering independent of the attacker strategy. We have demonstrated the robustness and practicality of Stamper by evaluating its performance using extensive data gathered from two widely used crowd computing systems, namely Yelp and Twitter.

Acknowledgements

We thank the anonymous reviewers and our shepherd, Jim Blomo, for their helpful comments. We also thank Arash Molavi Kakhki for his assistance with the Yelp dataset and Lisette Espín Noboa for her help with building the web user interface of the TrulyFollowing website. This research was supported in part by NSF grants CNS-1319019 and CNS-1421444. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

[6]Twitter periodically recommends a set of globally trending topics to users who are signed in to Twitter.

8. REFERENCES

[1] `http://en.wikipedia.org/wiki/Statistical_distance`.
[2] `http://tinyurl.com/guardian-cf-p`.
[3] `http://tinyurl.com/nyt-haggl`.
[4] `http://tinyurl.com/nyt-tw-sale`.
[5] `http://tinyurl.com/twitter-inactive`.
[6] `http://tinyurl.com/yelp-bought`.
[7] `http://tinyurl.com/yelp-consumer-alert`.
[8] `http://tinyurl.com/yelp-filter`.
[9] `http://tinyurl.com/yelp-filter-explained`.
[10] `http://tinyurl.com/yelp-halt`.
[11] `http://tinyurl.com/yelp-suspend`.
[12] F. Benevenuto, G. Magno, T. Rodrigues, and V. Almeida. Detecting spammers on twitter. In *Proceedings of the 7th Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference (CEAS)*, 2010.
[13] A. Beutel, W. Xu, V. Guruswami, C. Palow, and C. Faloutsos. Copycatch: stopping group attacks by spotting lockstep behavior in social networks. In *Proceedings of the 22nd international conference on World Wide Web (WWW)*, 2013.
[14] NIST/SEMATECH e-Handbook of Statistical Methods. `http://www.itl.nist.gov/div898/handbook/`.
[15] Q. Cao, M. Sirivianos, X. Yang, and T. Pregueiro. 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)*, 2012.
[16] Q. Cao, X. Yang, J. Yu, and C. Palow. Uncovering large groups of active malicious accounts in online social networks. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2014.
[17] R. T. Clemen and R. L. Winkler. Combining probability distributions from experts in risk analysis. *Risk analysis*, 19(2):187–203, 1999.
[18] J. Douceur. The Sybil Attack. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS)*, 2002.
[19] S. Feng, L. Xing, A. Gogar, and Y. Choi. Distributional footprints of deceptive product reviews. In *Proceedings of the the 6th International AAAI Conference on Weblogs and Social Media (ICWSM)*, 2012.
[20] M. Gabielkov and A. Legout. The complete picture of the twitter social graph. In *Proceedings of the 2012 ACM conference on CoNEXT student workshop*, 2012.
[21] S. Ghosh, B. Viswanath, F. Kooti, N. K. Sharma, G. Korlam, F. Benevenuto, N. Ganguly, and K. P. Gummadi. Understanding and combating link farming

in the twitter social network. In *Proceedings of the 21st International Conference on World Wide Web (WWW)*, 2012.

[22] V. J. Hodge and J. Austin. A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22(2):85–126, 2004.

[23] A. M. Kakhki, C. Kliman-Silver, and A. Mislove. Iolaus: Securing online content rating systems. In *Proceedings of the 22nd International World Wide Web Conference (WWW)*, 2013.

[24] S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[25] A. Lakhina, M. Crovella, and C. Diot. Diagnosing Network-wide Traffic Anomalies. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication (SIGCOMM*, 2004.

[26] E.-P. Lim, V.-A. Nguyen, N. Jindal, B. Liu, and H. W. Lauw. Detecting product review spammers using rating behaviors. In *Proceedings of the 19th ACM international conference on Information and knowledge management (CIKM)*, 2010.

[27] M. Motoyama, D. McCoy, K. Levchenko, S. Savage, and G. M. Voelker. Dirty jobs: The role of freelance labor in web service abuse. In *Proceedings of the 20th USENIX conference on Security (Usenix Security)*, 2011.

[28] J. X. Parreira, D. Donato, C. Castillo, and G. Weikum. Computing trusted authority scores in peer-to-peer web search networks. In *Proceedings of the 3rd International workshop on Adversarial information retrieval on the web*, 2007.

[29] B. C. Roy. *The Birth of a Word*. PhD thesis, MIT Media Lab, Feb 2013. `http://web.media.mit.edu/~bcroy/papers/bcroy-thesis_FINAL-sm.pdf`.

[30] R. R. Sillito and R. B. Fisher. Semi-supervised learning for anomalous trajectory detection. In *Proceedings of the British Machine Vision Conference 2008 (BMVC)*, 2008.

[31] G. Stringhini, M. Egele, C. Kruegel, and G. Vigna. Poultry markets: on the underground economy of twitter followers. In *Proceedings of the 2012 ACM workshop on Workshop on Online Social Networks*, 2012.

[32] G. Stringhini, G. Wang, M. Egele, C. Kruegel, G. Vigna, H. Zheng, and B. Y. Zhao. Follow the green: growth and dynamics in twitter follower markets. In *Proceedings of the 2013 conference on Internet measurement conference (IMC)*, 2013.

[33] K. Thomas, D. McCoy, C. Grier, A. Kolcz, and V. Paxson. Trafficking fraudulent accounts: The role of the underground market in twitter spam and abuse. In *Proceedings of the 22nd USENIX Security Symposium (USENIX Security)*, 2013.

[34] N. Tran, B. Min, J. Li, and L. Subramanian. Sybil-resilient online content voting. In *Proceedings of the 6th Symposium on Networked Systems Design and Implementation (NSDI)*, 2009.

[35] B. Viswanath, M. A. Bashir, M. Crovella, S. Guha, K. P. Gummadi, B. Krishnamurthy, and A. Mislove. Towards Detecting Anomalous User Behavior in Online Social Networks. In *Proceedings of the 23rd USENIX Security Symposium (Usenix Security)*.

[36] B. Viswanath, M. Mondal, A. Clement, P. Druschel, K. P. Gummadi, A. Mislove, and A. Post. Exploring the design space of social network-based Sybil defense. In *Proceedings of the 4th International Conference on Communication Systems and Network (COMSNETS)*, 2012.

[37] G. Wang, T. Konolige, C. Wilson, X. Wang, H. Zheng, and B. Y. Zhao. You Are How You Click: Clickstream Analysis for Sybil Detection. In *Proceedings of the 22nd USENIX Security Symposium (Usenix Security)*, 2013.

[38] G. Wang, T. Wang, H. Zheng, and B. Y. Zhao. Man vs. machine: Practical adversarial detection of malicious crowdsourcing workers. In *Proceedings of the 23rd USENIX Security Symposium (Usenix Security)*, 2014.

[39] G. Wang, C. Wilson, X. Zhao, Y. Zhu, M. Mohanlal, H. Zheng, and B. Y. Zhao. Serf and turf: crowdturfing for fun and profit. In *Proceedings of the 21st International conference on World Wide Web (WWW)*, 2012.

[40] G. Wu, D. Greene, B. Smyth, and P. Cunningham. Distortion as a validation criterion in the identification of suspicious reviews. In *Proceedings of the First Workshop on Social Media Analytics*, 2010.

[41] Z. Yang, C. Wilson, X. Wang, T. Gao, B. Y. Zhao, and Y. Dai. Uncovering social network Sybils in the wild. In *Proceedings of the 11th ACM/USENIX Internet Measurement Conference (IMC)*, 2011.

[42] H. Yu, C. Shi, M. Kaminsky, P. B. Gibbons, and F. Xiao. DSybil: Optimal sybil-resistance for recommendation systems. In *Proceedings of the 2009 30th IEEE Symposium on Security and Privacy (IEEE S&P)*, 2009.