Towards De-Anonymization of Google Play Search Rank Fraud

Mizanur Rahman, Nestor Hernandez, Bogdan Carbunar, Duen Horng Chau

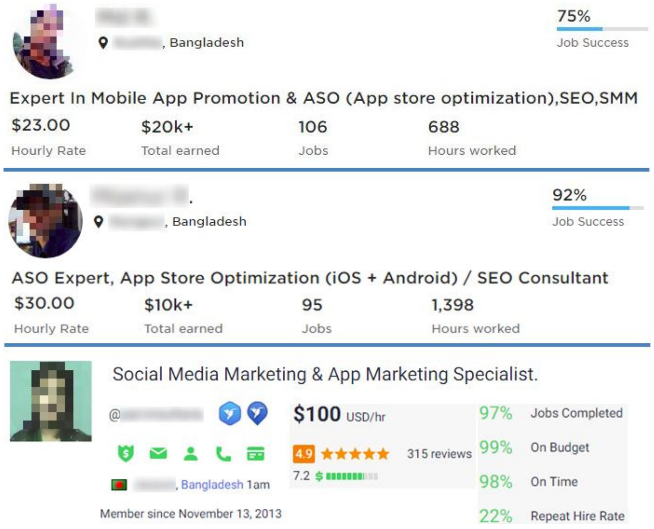
Abstract—Search rank fraud, the fraudulent promotion of products hosted on peer-review sites, is driven by expert workers recruited online, often from crowdsourcing sites. In this paper we introduce the *fraud de-anonymization* problem, that goes beyond fraud detection, to unmask the human masterminds responsible for posting search rank fraud in peer-review sites. We collect and study data from crowdsourced search rank fraud jobs, and survey the capabilities and behaviors of 58 search rank fraud workers recruited from 6 crowdsourcing sites. We collect a gold standard dataset of Google Play user accounts attributed to 23 crowdsourced workers and analyze their fraudulent behaviors in the wild. We propose DOLOS, a fraud de-anonymization system that leverages traits and behaviors we extract from our studies, to attribute detected fraud to crowdsourcing site workers, thus to real identities and bank accounts. We introduce MCDense, a min-cut dense component detection algorithm to uncover groups of user accounts controlled by *different* workers, and use stylometry and supervised learning to attribute them to crowdsourcing site profiles.

DOLOS correctly identified the owners of 95% of fraud worker-controlled communities, and uncovered fraud workers who promoted as many as 97.5% of fraud apps we collected from Google Play. When evaluated on 13,087 apps (820,760 reviews), which we monitored over more than 6 months, DOLOS identified 1,056 apps with suspicious reviewer groups. We report orthogonal evidence of their fraud, including fraud duplicates and fraud re-posts. DOLOS significantly outperformed adapted dense subgraph detection and loopy belief propagation competitors, on two new coverage scores that measure the quality of detected community partitions.

Index Terms—Search rank fraud, Peer-review system

1 INTRODUCTION

The developers of top ranking products in peer-review sites like Google Play, Amazon, or Yelp receive higher rewards, that include direct payments and ad-based revenues. Statistics maintained by peer-review sites concerning user activities for a product (e.g., reviews, ratings, likes, followers, app install counts) are known to play an essential part in the product's ranking [1], [2], [3]. This has created a black market for *search rank fraud*, mediated by an abundance of crowdsourcing sites, e.g., [4], [5], [6], [7], [8]. Specifically, crowdsourcing fraud workers create or purchase hundreds of user accounts in the peer-review site, then post activities for the products of developers who hire them, from the accounts they control, see Figure 1.



[redacted]
[redacted], Bangladesh
75%
Job Success

Expert In Mobile App Promotion & ASO (App store optimization),SEO,SMM

$23.00	$20k+	106	688
Hourly Rate	Total earned	Jobs	Hours worked

[redacted].
[redacted], Bangladesh
92%
Job Success

ASO Expert, App Store Optimization (iOS + Android) / SEO Consultant

$30.00	$10k+	95	1,398
Hourly Rate	Total earned	Jobs	Hours worked

Social Media Marketing & App Marketing Specialist.

@[redacted]
[redacted], Bangladesh 1am
Member since November 13, 2013

$100 USD/hr
4.9 ★★★★★ 315 reviews
7.2 $

97%	Jobs Completed
99%	On Budget
98%	On Time
22%	Repeat Hire Rate

Fig. 1. Anonymized snapshots of profiles of search rank fraud workers from Upwork (top 2) and Freelancer (bottom). Workers control hundreds of user accounts and earn thousands of dollars through hundreds of work hours. Our goal is to de-anonymize fraud, i.e., attribute fraud detected for products in online systems, to the crowdsourcing site accounts of the workers (such as these) who posted it.

Discouraging search rank fraud is essential to ensure trust in peer-review sites and the products that they host. Previous work in this area has focused on fraud detection [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21]. Most peer-review sites filter out detected fraudulent activities [22], [23], [24]. However, a study with 58 fraud workers that we recruited from 6 crowdsourcing sites revealed that workers with years of search rank fraud expertise are actively contributing to such jobs, and are able to post hundreds of reviews for a single product at prices ranging from a few cents to $10 per review. This suggests that fraud detection alone is unable to prevent large scale search rank fraud.

In this paper we propose a new approach to discourage search rank fraud. We introduce the *fraud de-anonymization* problem, that aims to attribute detected search rank fraud in a peer-review site, to the crowdsourcing site fraud workers who posted it. Further, to understand and model search rank fraud behaviors, we have developed a questionnaire and used it to survey 58 fraud workers recruited from 6 crowdsourcing sites. We have collected data from search

- *Mizanur Rahman, Nestor Hernandez and Bogdan Carbunar are with FIU. Email: {mrahm031, nhern121, carbunar}@cs.fiu.edu*
- *Duen Horng Chau is with Georgia Tech. Email: polo@gatech.edu*
- *A preliminary version of this article appears in ACM Hypertext 2018.*

rank fraud jobs and worker accounts in Upwork. We have collected a gold standard dataset of 956 user accounts in Google Play, attributed to 23 crowdsourced workers. We have developed a guilt-by-association process to expand this dataset with another 1,308 user accounts, for a total of 2,664 fraud worker-attributed accounts. We analyze the activities performed from these accounts in the wild. We observe and report several adversary traits, including the existence of an *expert core* of fraud workers, who can control hundreds of user accounts and post tens of daily reviews for a single product, can change their behaviors to avoid detection (e.g., to throttle their daily review activities and dilute them over long time intervals), can be rehired to promote the same product at later times, and that products can be fraudulently promoted by multiple workers.

We leverage the identified traits to introduce Dolos[1] a system that cracks down fraud by unmasking the human masterminds responsible for posting significant fraud. Dolos integrates search rank fraud detection with *fraud attribution* to reveal the lurking organized activities that power the fraud, and pinpoint their human command centers. Dolos detects then attributes fraudulent user accounts in the online service, to the crowdsourcing site accounts of the workers who control them. We devise MCDense, a min-cut dense component detection algorithm that analyzes common activity relationships between user accounts to uncover groups of accounts, each group controlled by a different search rank fraud worker. We then use stylometry and supervised learning to attribute MCDense detected groups to the crowdsourcing workers who control them.

Dolos correctly attributed 95% of the reviews of 640 apps (that received significant, ground truth search rank fraud) to their authors. For 97.5% of the apps, Dolos correctly de-anonymized at least one of the workers who authored their fake reviews. Dolos achieved 90% precision and 89% recall when attributing the above 2,664 fraudulent accounts to the workers who control them. Further, MCDense significantly outperformed an adapted densest subgraph solution.

We have evaluated Dolos on 13,087 Google Play apps (and their 820,760 reviews) that we monitored over more than 6 months. Dolos discovered that 1,056 of these apps have suspicious reviewer groups. Upon close inspection we found that (1) 29.9% of their reviews were *duplicates* and (2) 73% of the apps that had at least one MCDense-discovered clique, received reviews from the expert core fraud workers mentioned above. We also report cases of *fraud re-posters*, accounts who re-post their reviews, hours to days after Google Play filters them out (up to 37 times in one case).

To evaluate MCDense, we introduce two coverage scores, p-coverage and p-SCC, that measure the quality of detected community partitions. We adapt dense subgraph detection [25] and loopy believe propagation [18] solutions to the fraud de-anonymization problem. We show that MCDense consistently and significantly outperforms DSG on both coverage scores. While LBP can be used to accurately detect fraud, it cannot determine if all the accounts detected as fraudulent are controlled by a single or multiple workers.

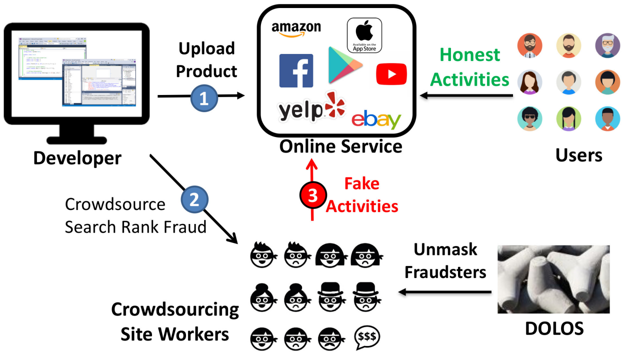


Fig. 2. **System and adversary model**. Developers upload products, on which users post activities. Adversarial developers crowdsource search rank fraud. Unlike fraud detection solutions, **Dolos unmasks the human workers responsible for posting search rank fraud**.

In summary, we introduce the following contributions:

- **Fraud de-anonymization problem formulation**. Introduce a new approach to combat and discourage search rank fraud in peer-review sites.
- **Study and model search rank fraud**. Survey 58 fraud workers from 6 crowdsourcing websites on fraud posting capabilities and behaviors. Collect search rank fraud jobs posted on Upwork and analyze common bidding and winning behaviors between workers. Collect gold standard fraud worker-attributed Google Play user accounts and study their behaviors in the real world. Extract and present fraud worker behaviors traits.
- **Dolos**. Exploit extracted insights to develop the first fraud de-anonymization system. Devise MCDense, a min-cut dense component detection algorithm to identify accounts controlled by the same worker. Use stylometry to attribute detected components to the profiles of known crowdsourcing workers.
- **Evaluation**. Evaluate Dolos extensively on Google Play data. Identify orthogonal evidence of fraud from detected suspicious products. Develop novel community coverage scores. Show that MCDense significantly outperforms adapted dense subgraph and loopy believe propagation solutions, on the developed scores.
- **Open source**. The Dolos and MCDense code is available for download online [26].

2 System and Adversary Model

We consider an ecosystem that consists of peer-review sites and crowdsourcing sites. Peer-review sites host accounts for developers, products and users, see Figure 2. Developers use their accounts to upload products. Users post *activities* for products, e.g., reviews, ratings, likes, installs. Product accounts display these activities and statistics, while user accounts list the products on which users posted activities.

Users register mobile devices to their accounts, then install apps on them. Users can only review apps that they have previously installed. Reviews have a star rating (1-5) and a text component.

The survival of mobile apps in Google Play is contingent on their search rank. Higher ranked apps are installed more frequently and generate more revenue, either through ads or direct payments. While Google keeps their ranking

[1] Dolos is a concrete block used to protect harbor walls from erosive ocean waves.

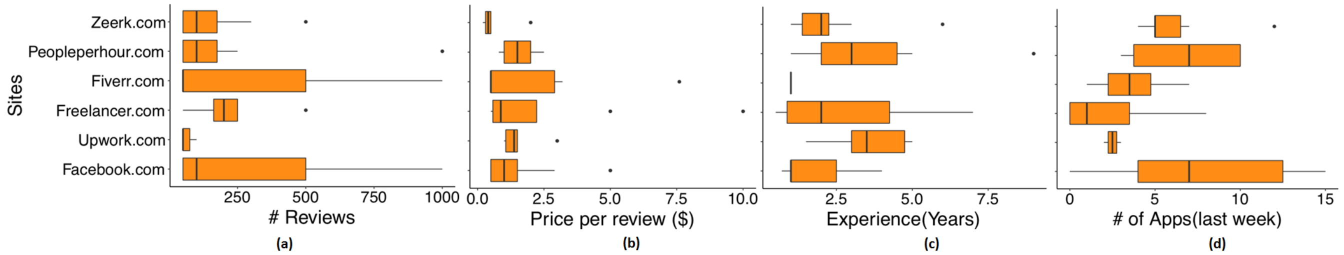


Fig. 3. Statistics over 44 fraud workers: minimum, average and maximum for (a) number of reviews that a worker can write for an app, (b) price demanded per review, (c) years of experience, (d) number of apps reviewed in the past 7 days. Workers report to be able to write hundreds of reviews for a single app, have years of experience and are currently active. Prices range from 56 cents to $10 per review.

algorithm secret, popular belief (e.g., [3]) holds that large numbers of positive reviews help new apps achieve higher search rank.

Crowdsourcing sites host accounts for *workers* and *employers*. Worker accounts have unique identifiers and bank account numbers used to deposit the money that they earn. Employers post *jobs*, while workers *bid* on jobs, and, following negotiation steps, are assigned or *win* the jobs.

We consider product developers who hire workers from crowdsourcing sites, to perform search rank fraud, see Figure 1. In this paper we focus on workers who control multiple user accounts in the online system, which they use to post fake activities, e.g., review, rate, install. We study such workers in § 4.

3 The Fraud De-Anonymization Problem

Let $\mathcal{W} = \{W_1, .., W_n\}$ be the set of crowdsourcing worker accounts. Let $\mathcal{U} = \{U_1, .., U_m\}$ be the set of user accounts and let $\mathcal{A} = \{A_1, .., A_a\}$ be the set of products hosted by the online service, respectively. We define the fraud de-anonymization problem as follows:

Fraud De-Anonymization Problem. Given a product $A \in \mathcal{A}$, return the subset of fraud workers in $\mathcal{W}$ who control user accounts in $\mathcal{U}$ that posted activities for A.

Unlike standard de-anonymization, which refers to the adversarial process of identifying users from data where their Personally Identifiable Information (PII) has been removed, the fraud de-anonymization problem seeks to attribute detected search rank fraud to the humans who posted it.

A solution to this problem will enable peer-review sites to (1) put a face to the humans who post fraud for the products that they host, i.e., identify their banking information and use it to pursue fraud workers, and (2) provide proof of fraud to customers, e.g., through links to the crowdsourcing accounts responsible for fraud posted on products they browse, see Figure 1. Thus, fraud de-anonymization may provide counter-incentives both for the crowdsourcing workers who participate in fraud jobs, and for the product developers who recruit fraud workers.

4 A Study of Search Rank Fraud

We now describe our efforts to understand and model fraud workers. Succinctly, we have (1) performed a user study with fraud workers recruited from several crowdsourcing sites, (2) collected and analyzed search rank fraud data from Upwork, (3) collected a gold standard set of user accounts, attributed to a worker identified from a crowdsourcing site and (4) analyzed the behaviors exhibited by these user accounts. We have developed our protocols to interact with participants and collect data in an IRB-approved manner (Approval #: IRB-15-0219@FIU and IRB-18-0077@FIU). In the following we describe each contribution.

4.1 Motivation: Fraud Worker Capabilities

To evaluate the magnitude of the problem, we have first contacted 44 workers from several crowdsourcing sites including Zeerk (12), Peopleperhour (9), Freelancer (8), Upwork (6) and Facebook groups (9), who advertised search rank fraud capabilities for app markets. We asked them (1) how many reviews they can write for one app, (2) how much they charge for one review, (3) how many apps they reviewed in the past 7 days, and (4) for how long they been active in promoting apps.

Figure 3 shows statistics over the answers, organized by crowdsourcing site. It suggests significant profits for fraud workers, who claim to be able to write hundreds of reviews per app (e.g., an average of 250 reviews by Freelancer workers) and charge from a few cents ($0.56 on average from Zeerk.com workers) to $10 per review (Freelancer.com). Workers have varied degrees of expertise in terms of years of experience and recent participation in fraud jobs. For instance, in recently emerged Facebook groups, that either directly sell reviews or exchange reviews, workers have less than 2.5 years experience, but are very active, with more than 7 jobs in the past 7 days on average, and are economical ($1.3 on average per review). Further, workers from Peopleperhour and Upwork have more than 2.5 years experience and more than 3 recent jobs on average.

Subsequently, we have developed a more detailed questionnaire to better understand search rank fraud behaviors and delivered it to 14 fraud freelancers that we recruited from Fiverr. We paid each participant $10, for a job that takes approx. 10 minutes. The IPs from which the questionnaire was accessed revealed that the participants were from Bangladesh (5 participants), USA (2), Egypt (2), Netherlands, UK, Pakistan, India and Germany (1). The participants declared to be male, 18 - 28 years old, with diverse education levels: less than high school (1 participant), high school (2), associate degree (3), in college (5), bachelor degree or more (3).

The participants admitted an array of fraud expertise (fake reviews and ratings in Google Play, iTunes, Amazon, Facebook and Twitter, fake installs in Google Play and iTunes, fake likes and followers in Facebook and Instagram,

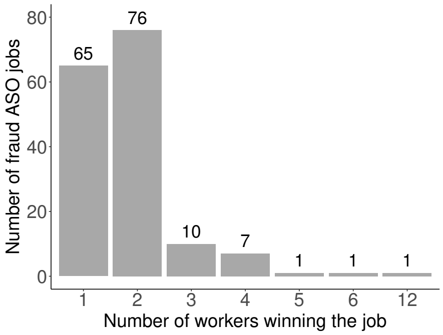


Fig. 4. Distribution of winning workers for search rank fraud jobs: developers hire multiple workers. More jobs are assigned to 2 or more workers than to 1. This reveals the need for DOLOS to detect fraudulent communities and attribute them to different fraud workers.

influential tweets in Twitter). With a focus on search rank fraud targeting Google Play apps, we found a mix of (1) inexperienced and experienced fraud workers: 4 out of 14 had been active less than 2 months and 6 workers had been active for more than 1 year, and (2) active and inactive workers: 4 had not worked in the past month, 9 had worked on 1-5 fraud jobs in the past month, and 1 worked on more than 10 jobs; 8 workers were currently active on 1-5 fraud jobs, and 1 on more than 5.

Further, we observed varying search rank fraud capabilities when it comes to the magnitude of the fraud on a per-app level. For instance, 1 worker wrote at most 1 review per app, 2 wrote 2-5 reviews, 7 workers said that they wrote between 5 to 50 reviews per app, while 1 wrote 51 to 100 reviews. 1 worker performed less than 10 installs per job, 3 had 11 to 100 installs, 3 had 101 to 1,000 installs per job, while 1 worker said that he performed more than 1,000 installs for a single app. 8 workers claimed access to more than 10 mobile devices, with 1 having more than 50.

Of the 14 fraud workers surveyed, 3 admitted to working in teams that had more than 10 members, and to sharing the user accounts that they control, with others. 10 workers said that they control more than 5 Google Play accounts and 1 worker had more than 100 accounts. Later in this section we show that this is realistic, as other 23 workers we recruited, were able to reveal between 22 and 86 Google Play accounts that they control. Further, 4 workers said that they never abandon an account, 5 said that they use each account until they are unable to login, and 4 said that they use it for at most 1 year. This is confirmed by our empirical observation of the persistence of fraud (see end of section 4.3).

4.2 A Study of Search Rank Fraud Jobs

We identified and collected data from 161 search rank fraud jobs in Upwork that request workers to post reviews on, or install Google Play and iTunes apps. We have collected the 533 workers who have bid on these jobs. We call the bidding workers that are awarded a job, *winners*. To achieve this, we have developed a Python crawler to collect data both from crowdsourcing sites and from Google Play.

Figure 4 shows the distribution of the number of winners per search rank fraud job. One job of the 161, was awarded to 12 workers; more jobs were awarded to 2 workers than to only 1. This indicates that hiring multiple workers is considered beneficial by adversarial developers, and suggests the need to attribute detected organized fraud activities to human masterminds (see next section).

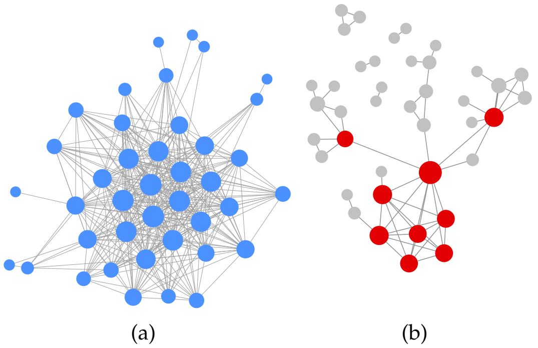

(a) (b)

Fig. 5. (a) **Worker co-bid graph**: Nodes are Upwork workers. An edge connects two workers who co-bid on search rank fraud jobs. We see a tight co-bid community of workers; some co-bid on 37 jobs. (b) **Worker Co-win graph** with an “expert core” of 8 workers (red), each winning 8-15 jobs. Edges connect workers who won at least one job together. Any two workers collaborated infrequently, up to 4 jobs.

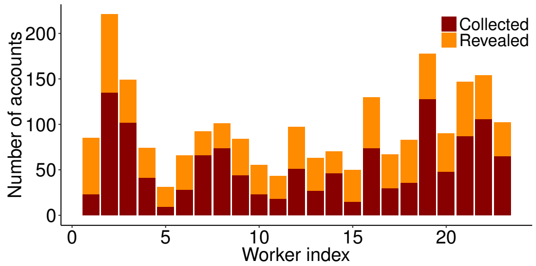


Fig. 6. **Attributed, fraud worker-controlled accounts.** The numbers of Google Play accounts revealed by the workers are shown in red. Each worker has revealed 22-86 accounts. Guilt-by-association accounts are shown in orange. We have collected a total of 2,664 accounts (red + orange). One worker controls (at least) 217 accounts.

In order to understand the extent to which crowdsourced workers participate in common on search rank fraud jobs, we introduce the concepts of *co-bid* and *co-win graphs*. In the co-bid graph, nodes are workers who bid on fraud jobs; edges connect workers who bid together on at least one job. The edge weights denote the number of jobs on which the endpoint workers have bid together. In the co-win graph, the weight of an edge is the number of fraud jobs won by both endpoint workers.

Out of the 56 workers who won the 161 jobs, only 40 had won a job along with another bidder. Figure 5(a) shows the co-bid graph of these 40 winners, who form a tight community. Figure 5(b) plots the co-win graph of the 40 winners. We observe an “expert core” of 8 workers who each won between 8 to 15 jobs. Further, we observe infrequent collaborations between any pair of workers: any two workers collaborated on at most 4 jobs.

4.3 Fraud Worker Profile Collection (FPC)

We have collected a first gold standard dataset of attributed, fraud worker controlled accounts in Google Play. For this, we have identified and contacted 100 Upwork, Fiverr and Freelancer workers with significant bidding activity on

TABLE 1
Account attribution performance on gold standard fraud worker-controlled dataset, with several supervised learning algorithms (parameters $d = 300$, $t = 100$, $\gamma = 80$, and $w = 5$ set through a grid search). SVM performed best.

Algorithm	Precision	Recall	F-measure
RF	95.5%	91.6%	93.5%
SVM	**98.5%**	**98.3%**	**98.5%**
k-NN	97.1%	96.4%	96.7%
MLP	98.6%	98.1%	98.4%

search rank fraud jobs targeting Google Play apps. Figure 6 shows the number of accounts (bottom, red segments) revealed by each of 23 most responsive of these workers: between 22 and 86 Google Play accounts revealed per worker, for a total of 956 user accounts. We call this dataset "gold standard", because it is not ground truth: we do not have complete confidence that the workers do indeed control all of the accounts [2]. However, this is a first effort to collect attributed fraud data that balances the need to involve the fraud workers in this process, with the need to satisfy ethical and site terms-of-service constraints.

Fraud app dataset. To expand this data, we collected first a subset of 640 apps that received the highest ratio of reviews from accounts controlled by the above 23 expert core workers to the total number of reviews. We have monitored the apps over a 6 months interval, collecting their new reviews once every 2 days. The 640 apps had between 7 to 3,889 reviews. Half of these apps had at least 51% of their reviews written from accounts controlled by the 23 fraud workers. In the following we refer to these, as the *fraud apps*.

Union fraud graph. We have collected the account data of the 38,123 unique reviewers (956 of which are the seed accounts revealed by the 23 fraud workers) of the fraud apps, enabling us to build their *union fraud graph*: a node corresponds to an account that reviewed one of these apps (including worker controlled and honest ones), and the weight of an edge denotes the number of apps reviewed in common by the accounts that correspond to the end nodes. We have removed duplicates: an account that reviewed multiple fraud apps has only one node in the graph. The union fraud graph has 19,375,550 edges and 162 disconnected components, of which the largest has 37,566 nodes.

Guilt-by-association. We have labeled each node of the union fraud graph with the ID of the worker controlling it or with "unknown" if no such information exists. For each unknown node U, we decide if U is controlled by one of the workers, based on how well U is associated with accounts controlled by the worker. However, U may be connected to the accounts of multiple workers (Trait 3, § 4.5).

To address this problem, we leveraged Trait 4 (see § 4.5) to observe that random walks that start from nodes controlled by the same fraud workers are likely to share significant context, likely different from the context of nodes controlled by other workers, or that are honest. We have pre-processed the union fraud graph to convert it into a non-weighted graph: replace an edge between nodes u_i and u_j with weight w_{ij}, by w_{ij} non-weighted edges between u_i and u_j. We then used the DeepWalk algorithm [27] to perform γ random walks starting from each node v in this graph, where a walk samples uniformly from the neighbors of the last vertex visited until it reaches the maximum walk length (t). The pre-processing of the union graph ensures that the probability of DeepWalk at node u_i to choose node u_j as next hop, is proportional to w_{ij}. DeepWalk also takes as input a window size w, the number of neighbors used as the context in each iteration of its SkipGram [27] component. Deepwalk returns a d-dimensional representation in $\mathbb{R}^d$ for each of the nodes. We then used this representation as predictor features for the "ownership" of the account U, i.e., the worker who controls it.

Table 1 highlights precision, recall, and F-measure achieved by different supervised learning algorithms. We observe that SVM reaches 98.5% F-measure which suggests DeepWalk's ability to provide useful features and assist in our guilt-by-association process. We then applied the trained model to the remaining and unlabeled accounts in the union fraud graph obtaining new guilt-by-association accounts for each of the 23 workers. Figure 6 shows the number of seed and guilt-by-association accounts uncovered for each of the 23 workers. We have collected $1,708$ additional accounts, for a total of 2,664 accounts.

Persistence of fraud. After more than 1 year following the collection of the 2,664 fraud worker-controlled accounts, we have re-accessed the accounts. We found that 67 accounts had been deleted and 529 accounts were inactive, i.e., all information about apps installed, reviewed, +1'd was removed. 2,068 accounts were active. This is consistent with the findings from our worker survey, where 4 out of 14 surveyed workers said that they never abandon an account, 5 said that they use each account until they are unable to login, and 4 said that they use it for at most 1 year. This further suggests the limited ability of Google Play to identify and block worker-controlled accounts.

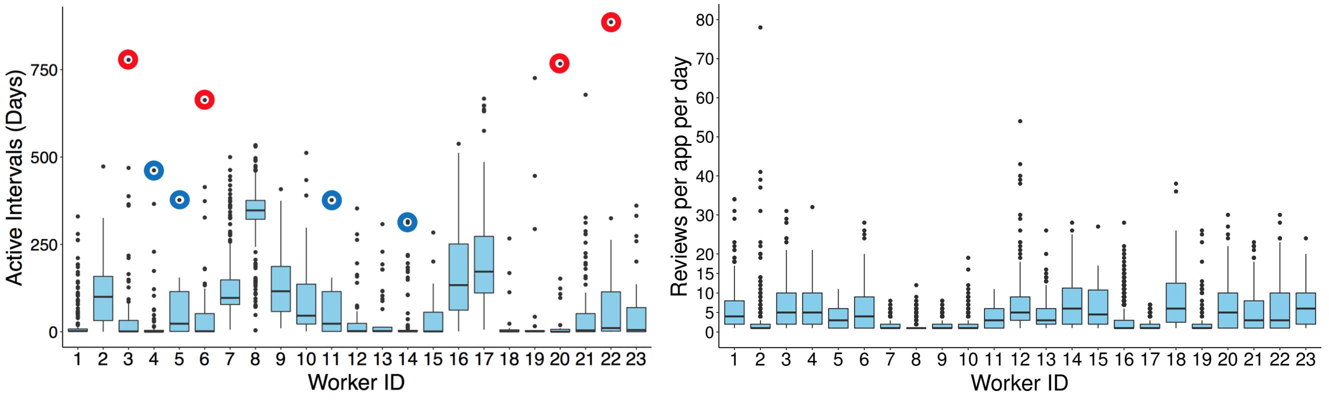
4.4 Analysis of Fraud Behaviors

We study the activities performed from the accounts controlled by the above 23 fraud workers, in Google Play. For this, we have selected the 2,835 apps that have received at least 10 reviews from the 2,664 accounts controlled by the 23 workers. We perform our analysis on these apps.

Active intervals. First, we study the *active interval* length of a worker for an app: the time interval between the first and last reviews posted from accounts controlled by the worker, for the app. Figure 7(a) shows the per-worker distribution of active interval durations, for the above 2,835 apps. We observe several apps (e.g., shown with red and blue circles) that were targeted by each of several workers, over long time intervals (e.g., 1-2 years). We posit that workers may be rehired several times over the years, to perform search rank fraud. We revisit this hypothesis shortly.

Daily review capabilities. Second, we study the number of reviews that a worker has been able to post in a single day for a single app, from all the user accounts it controls. Figure 7(b) shows the distribution of the number of daily reviews posted by each of 23 fraud workers, per each app they target. It shows that several workers had days when they were able to post more than 38 reviews per day for one app. The second worker posted 78 reviews in a day for one

[2] We have however verified that the workers knew the Gmail address associated with each account



(a)

(b)

Fig. 7. (a) Per-fraud worker distribution of active intervals. Each point denotes the length of the active interval of the corresponding worker for a Google Play app that he has targeted. The red dots correspond to a Google Play app, while the blue dots correspond to another app. Each of these apps was targeted by 4 of the 23 workers. (b) Distributions of number of daily reviews posted by each worker per app. Workers 2, 12, 43 and 18 provide an average of 38 to 78 daily reviews per each app they have targeted.

app! These results corroborate the findings of our worker survey described in § 4.1.

Active intervals vs. reviews per active day. We found, using 3,369 (app, fraud worker) data points, that the fraud workers who post a high number of reviews on average per active day (e.g., 18-34), tend to target apps only for a short time span (small active interval length), i.e., over 1-2 days. However, these points account for only 1.6% of the data. 75% of the data points plotted correspond to active intervals of up to 250 days. 64% of these points correspond to (app, fraud worker) pairs where the worker wrote an average of 1 - 3 reviews per active day, 18% to 4 - 7 reviews per day and 18% to 18-34 reviews per day. 25% (858) of the data points correspond to active intervals of between 250 and 887 days. 72.12% of the points correspond to (app, fraud worker) pairs where the worker wrote an average of 1 - 3 reviews per active day.

We further observed that search rank fraud workers often "dilute" their reviews over a large number of days, instead of posting large number of reviews over only a few days. We believe that this is due to job requirements, which have evolved to avoid obvious defenses employed by peer-review systems, e.g., through detection of review spikes.

4.5 Empirical Adversary Traits

We summarize now several search rank fraud worker traits suggested by our studies:

- **Trait 1**: Fraud workers control multiple user accounts which they use to perpetrate search rank fraud.
- **Trait 2**: While workers have diverse search rank fraud capabilities, crowdsourcing sites have an "expert core" of successful workers. Many fraud workers are willing to contribute, but few have the expertise or reputation to win such jobs.
- **Trait 3**: Search rank fraud jobs often recruit multiple workers. Thus, targeted products may receive fake reviews from multiple workers. This suggests that in addition to identifying fraudulent reviews, we need to further attribute them to their authors.
- **Trait 4**: Any two fraud workers collaborate infrequently, when compared to the number of search rank fraud jobs on which they have participated, see Figure 5(b).
- **Trait 5**: Fraud workers and the people who hire them evolve their strategies, to avoid detection.
- **Trait 6**: Fraud workers may be rehired by the same product developer to promote the same product, several times over the years.
- **Trait 7**: Fraud workers, including experts, are willing to reveal information about their behaviors, perhaps to convince prospective employers of their expertise.

DOLOS exploits these traits to detect and attribute groups of fraudulent user accounts to the fraud workers who control them. We do not claim that the sample data from which the traits are extracted is representative. However, in the evaluation section we show that DOLOS can accurately de-anonymize fraud workers.

5 FRAUD DE-ANONYMIZATION SYSTEM

5.1 Solution Overview

We introduce DOLOS, the first fraud de-anonymization system that integrates activities on both crowdsourcing sites and online services. As illustrated in Figure 8, DOLOS performs two tasks: (Task 1) proactively identify new fraud workers and builds their profiles in crowdsourcing sites, and (Task 2) process product and user accounts in online systems to attribute detected fraud to these profiles. The FPC module described in the previous section performs Task 1. In the following, we focus on Task 2, which we break into two sub-problems:

- **Fraud-Component Detection Problem**. Given a product $A \in \mathcal{A}$, return a set of components $C_A = \{C_1, .., C_k\}$, where any $C_{j=1..k}$ consists of a subset of the user accounts who posted an activity for A, s.t., those accounts are either controlled by a single worker in $\mathcal{W}$, or are honest.
- **Component Attribution Problem**. Given $\mathcal{W}$ and a component $C \in C_A$, return the identity of the worker in $\mathcal{W}$ who controls all the accounts in the component, or $\perp$ if the accounts are not controlled by a worker.

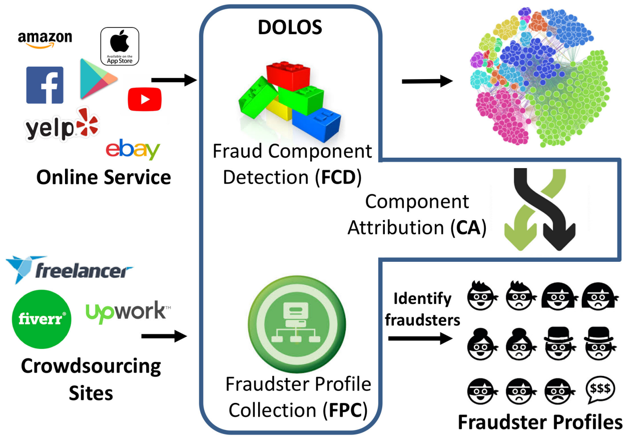


Fig. 8. **DOLOS system architecture**. The Fraud Component Detection (FCD) module partitions the co-activity graphs of apps into loosely inter-connected, dense components. The Component Attribution (CA) module attributes FCD detected components to worker profiles collected by the worker Profile Collector (FPC).

Algorithm 1 MCDense: Min-Cut based Dense component detection. We set η to 5 and tau to 0.5.

```
Input: G = $(\mathcal{U}, \mathcal{E}_w)$: input graph
        n := $|\mathcal{U}|$
Output: $\mathcal{C} := \emptyset$: set of node components
1. MCDense(G){
2.     if (nodeCount(G) $< \eta$) return;
3.     $(G_1, G_2)$ := weightMinCut(G);
4.     if (($\rho(G_1) > \rho(G)$ & $\rho(G_2) > \rho(G)$)
           & ($\rho(G) < \tau$)){
5.         MCDense($G_1$); MCDense($G_2$);
6.     else
7.         $\mathcal{C} := \mathcal{C} \cup$ G;
8.         return;
9.     end if
```

DOLOS's FCD and CA modules respectively, provide solutions to these sub-problems. In the following, we detail the FCD and CA modules.

5.2 Fraud Component Detection (FCD) Module

In order to identify communities, each controlled by a different fraud worker, we leverage the adversary Trait 4, that the accounts controlled by one worker are likely to have reviewed significantly more products in common than with the accounts controlled by another worker. We introduce *MCDense*, an algorithm that takes as input the **co-activity graph** of a product A, and outputs its *fraud components*, sets of user accounts, each potentially controlled by a different worker. We define the **co-activity graph** of a product A as $G = (\mathcal{U}, \mathcal{E}_w)$, with a node for each user account that posted an activity for A (see Figure 9 for an illustration). Two nodes $u_i, u_j \in \mathcal{U}$ are connected by a weighted edge $e(u_i, u_j, w_{ij}) \in \mathcal{E}^w$, where the weight w_{ij} is the number of products on which u_i and u_j posted activities in common.

MCDense, see Algorithm 1, detects densely connected subgraphs, each subgraph minimally connected to the other subgraphs. Given a graph $G = (\mathcal{U}, \mathcal{E}_w)$, its triangle density is $\rho(G) = \frac{t(V)}{\binom{|V|}{3}}$, where $t(V)$ is the number of triangles formed by the edges in $\mathcal{E}^w$.This definition differs from Tsourakakis [25] in the DSG algorithm (see § 6.4.1). Thus, unlike ρ_D that can be larger than 1, $\rho \in [0, 1]$.

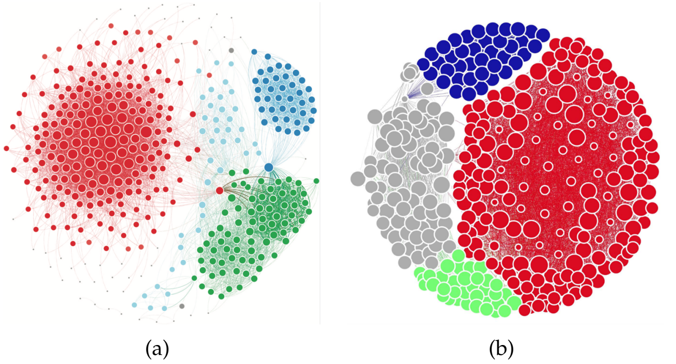


Fig. 9. **Co-review graph** of user accounts reviewing a popular horoscope app in Google Play (name hidden for privacy). Nodes are accounts. 4 Upwork workers each revealed to control the accounts of the same color. Two accounts are connected if they post reviews for the same apps. Node sizes are a function of the account connectivity. (b) DOLOS found these 4 tightly connected groups of accounts, and **correctly attributed 3 groups** to the workers controlling them.

MCDense recursively divides the co-activity graph into two minimally connected subgraphs: the sum of the weights of the edges crossing the two subgraphs, is minimized. If both subgraphs are more densely connected than the initial graph (line 4) and the density of the initial graph is below a threshold τ, MCDense treats each subgraph as being controlled by different workers: it calls itself recursively for each subgraph (lines 5,6). Otherwise, MCDense considers the undivided graph to be controlled by a single worker, and adds it to the set of identified components (line 8).

We have used the gold standard set of accounts controlled by the 23 fraud workers detailed in the previous section, to empirically set the τ threshold to 0.5, as the lowest density of the 23 groups of accounts revealed by the workers was just above 0.5.

MCDense converges and has $O(|\mathcal{E}_w||\mathcal{U}|^3)$ complexity. To see that this is the case, we observe that at each step, MCDense either stops or, at the worst, "shaves" one node from G. The complexity follows then based on Karger's min-cut algorithm complexity [28].

5.3 Component Attribution (CA) Module

Given a set of fraud worker profiles $\mathcal{FW}$ and a set of fraud components returned by the FCD module for a product A, the component attribution module identifies the workers likely to control the accounts in each component. To achieve this, DOLOS leverages the unique writing style of human workers to fuse elements from computational linguistics, e.g., [29], [30], and author de-anonymization, e.g., [31]. Specifically, we propose the following 2-step component attribution process:

CA Training. Identify the products reviewed by the accounts controlled by each worker $W \in \mathcal{FW}$. For each such product, create a *review instance* that consist of all the reviews written by the accounts controlled by W for A. Thus, each review instance contains only (but all) the reviews written from the accounts controlled by a single worker, for a single product. Extract stylometry features from each review instance of each worker, including character count, average number of characters per word, and frequencies of letters, uppercase letters, special characters, punctuation marks, digits, numbers, top letter digrams, trigrams, part

Algorithm 2 DOLOS pseudocode. Given a set of crowdsourcing sites and peer-review site products, identify prolific fraud workers, accounts they control and targeted apps.

```
Input: Prod[ ] Products : monitored products
       site[ ] crowdSites : monitored sites
       int $\phi$ : threshold account number signal expertise
Output: $<$ F, Acc $>$[ ] workers : detected fraud
        Prod[ ] fraudProd : detected fraud products
1.  DOLOS (){
2.  while (true) do
3.      $\langle$ F, Acc $\rangle$[ ] fraud := FPC.getSeeds(crowdSites);
4.      candidates.add(fraud); CA.train(candidates);
5.      for each prod in Products do
6.          C := MCDense.getComponents(prod);
7.          if (C.size $\neq$ 0) then fraudProd.add(prod);
8.          for each c in C do
9.              F f := CA.attribute(c, candidates);
10.             UserAcc a := candidates.getAccounts(f);
11.             a.add(c.accounts);
12.     for each $\langle$ f, a $\rangle$ in candidates do
13.         if (a.size $\geq \phi$) then
14.             workers.add($\langle$ f, a $\rangle$);
15.             candidates.remove($\langle$ f, a $\rangle$); }
```

of speech (POS) tags, POS digrams, POS trigrams, word digrams, word trigrams and of misspelled words. Train a supervised learning algorithm on these features, that associates the feature values of each review instance to the worker who created it.

Attribution. Let $\mathcal{C}$ denote the set of components returned by MCDense for a product A. For each component $C \in \mathcal{C}$, group all the reviews written by the accounts in C for product A, into a review instance, r. Extract r's stylometry features and use the trained classifier to determine the probability that r was authored by each of the workers in $\mathcal{FW}$. Output the identity of the fraud worker with the highest probability of having authored r.

5.4 Putting It All Together

Algorithm 2 shows the pseudocode of DOLOS. DOLOS takes as input a list of crowdsourcing sites and a list of products, and generates a list of identified prolific fraud workers and accounts that they are suspected to control in the online service, along with a list of the products on which they have performed search rank fraud.

DOLOS uses FPC (see § 4.3) to identify a fresh set of seed fraud from crowdsourcing sites, that consists of a new set of crowdsourcing site workers F, along with a set of user accounts Acc that each worker controls in the peer-review site (Algorithm 2, line 3). It then adds this seed information to the set of candidate fraud workers and uses it to re-train the component attribution (CA) module (line 4).

For each product received in its input (line 5), DOLOS uses MCDense to find the densely connected components of its co-review graph (line 6). If it finds at least one such component, it adds the product to the list of products targeted by search rank fraud (line 7), then, for each component, it uses the trained CA module to attribute the accounts in the component (line 9), and adds the accounts to the list of accounts controlled by the identified fraud worker (lines 10-11). At the end of this process, DOLOS plucks the expert workers (i.e., who now control more than the threshold ϕ user accounts) and adds them to the list of workers that it outputs (lines 12-15).

DOLOS repeats the above steps each time FPC identifies more seed ground truth data (line 2).

6 EMPIRICAL EVALUATION

In this section we compare the results of DOLOS on fraud and honest apps, evaluate its de-anonymization accuracy, and present its results on 13,087 apps. Further, we compare MCDense with adapted dense sub-graph detection and loopy belief propagation solutions.

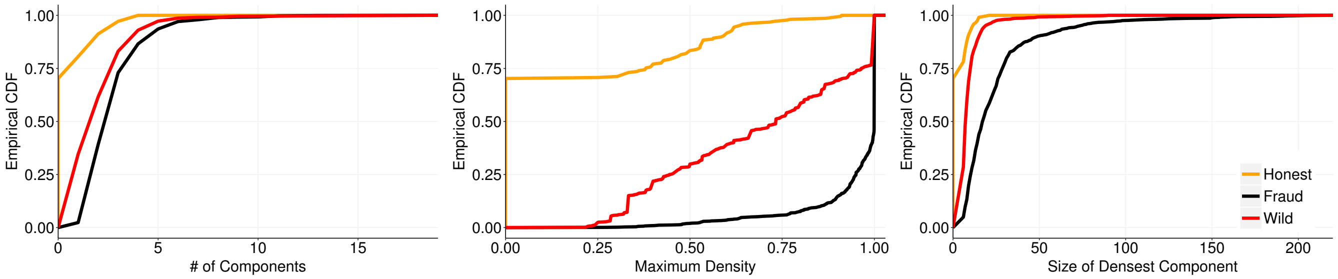
6.1 Fraud vs. Honest Apps

We evaluate the ability of DOLOS to discern differences between fraudulent and honest apps. For this, we first selected 925 candidate apps from the longitudinal app set, that have been developed by Google designated "top developers". We have filtered those flagged by VirusTotal. We have manually investigated the other apps, and selected 219 apps that (i) have more than 10 reviews and (ii) were developed by reputable media outlets (e.g., Google, PBS, Yahoo, Expedia, NBC) or have an associated business model (e.g., fitness trackers). We have collected 38,224 reviews and their associate user accounts from these apps.

Figure 10(a) compares the CDF of the number of components (of at least 5 accounts) found by MCDense per each of the 640 fraud apps vs. the 219 honest apps. MCDense found that all the fraud apps had at least 1 component, however, 70% of the honest apps had no component. The maximum number of components found for fraud apps is 19 vs. 4 for honest apps. Figure 10(b) compares the CDF of the maximum edge density (ratio of number of edges to maximum number of edges possible) of a component identified by MCDense per fraud vs. honest apps. 94.4% of fraud apps have density more than 75% while only 30% of the honest apps have a cluster with density larger than 0. The increase is slow, with 90% of the honest apps having clusters with density of 60% or below. Figure 10(c) compares the CDF of the size of the per-app densest component found for fraud vs. honest apps. 80% of the fraud apps vs. only 7% of the honest apps, have a densest component with more than 10 nodes. The largest, densest component has 220 accounts for a fraud app, and 21 accounts for an honest app. We have manually analyzed the largest, densest components found by MCDense for the honest apps and found that they occur for users who review popular apps such as the Google, Yahoo or Facebook clients, and users who share interests in, e.g., social apps or games.

6.2 De-Anonymization Performance

We have implemented the CA module using a combination of JStylo [32] and supervised learning algorithms. We have collected the 111,714 reviews posted from the 2,664 attributed, fraud worker controlled user accounts of § 4.3. The reviews were posted for 2,175 apps. We have grouped these reviews into instances, and we have filtered out those with less than 5 reviews. The remaining total is 6,046 instances, 40 to 1,664 per worker. Figure 11(a) shows their distribution among the 23 workers who authored them.



(a) (b) (c)

Fig. 10. MCDense: Cumulative distribution function (CDF) over 640 fraud, 219 honest, and 1,056 suspicious "wild" apps, of per-app (a) number of components of at least 5 accounts, (b) maximum density of an identified component and (c) size of densest component.

TABLE 2
DOLOS attribution performance for the 1,690 instances of the 640 fraud apps. k-NN identifies the workers responsible for 95% of the instances.

Algo	Top 1	(TPR)	Top 3	Top 5
k-NN (IBK)	**1608**	**(95.0%)**	**1645**	1646
RF (Random Forest)	1487	(87.9%)	1625	**1673**
DT (Decision Tree)	1126	(66.5%)	1391	1455
SVM	1101	65%	1195	1214
NB (Naive Bayes)	569	36.5%	874	1067
SMO	1117	68.3%	1434	1548

We have evaluated the performance of DOLOS (MCDense + CA) using a leave-one-out cross validation process [3] over the 640 fraud apps (and their 1,690 review instances). More specificaly, for each app A, CA extracts stylometric features from each review instance with JStylo [32] (see § 5.3), then trains a supervised learning algorithm on all the review instances minus the instances that were written for A. During testing, DOLOS converts each fraud component returned by MCDense for A into a review instance, that contains the reviews written by its accounts for A. It then uses the trained CA to determine the workers most likely to have authored it. Thus, DOLOS trains a different classifier for each test app.

We have used several supervised learning algorithms, including k-nearest neighbors (k-NN), Random Forest (RF), Decision Trees (DT), Naive Bayes (NB), Support Vector Machine (SVM), and Sequential Minimal Optimization (SMO).

Instance level performance. Table 2 shows the number of instances correctly attributed by DOLOS (out of the 1,690 instances of the 640 fraud apps) and corresponding true positive rate, as well as the number of instances where the correct worker is among DOLOS' top 3 and top 5 options. k-NN achieved the best performance, correctly identifying the workers responsible for posting 95% of the instances. We observe that k-NN correctly predicts the authors of 95% of the instances. Figure 11(b) zooms into per-fraud worker precision and recall, showing the ability of DOLOS to identify the instances and only the instances of each of the 23 workers. For 21 out of 23 workers, the DOLOS precision and recall both exceed 87%.

App level performance. Table 3 shows that when using k-NN, DOLOS correctly identified at least 1 worker per app, for 97.5% of the fraud apps, and identified at least 90% of the workers in each of 87% of the fraud apps. Table 4 shows that the precision of DOLOS in identifying an app's workers exceeds 90% for 69% of the apps.

TABLE 3
Number of apps for which DOLOS has a recall of at least 50%, 70% and 90%. k-NN identifies at least one worker for 97.5% of the 640 fraud apps, and 90% of the workers of each of 557 (87%) of the apps.

Algo	1 worker	50%-recall	70%-recall	90%-recall
RF	624	622	537	465
SVM	574	517	325	284
k-NN	**625**	**625**	**585**	**557**
DT	554	510	315	264
NB	379	256	128	125
SMO	533	492	318	265

TABLE 4
App level precision: the number of apps where its precision is at least 50%, 70% and 90%. The precision of DOLOS when using k-NN exceeds 90% for 69% of the fraud apps.

Algo	50%-prec	70%-prec	90%-prec
RF	573	434	359
SVM	460	249	209
k-NN	**578**	**483**	**444**
DT	446	260	208
NB	217	100	99
SMO	436	264	212

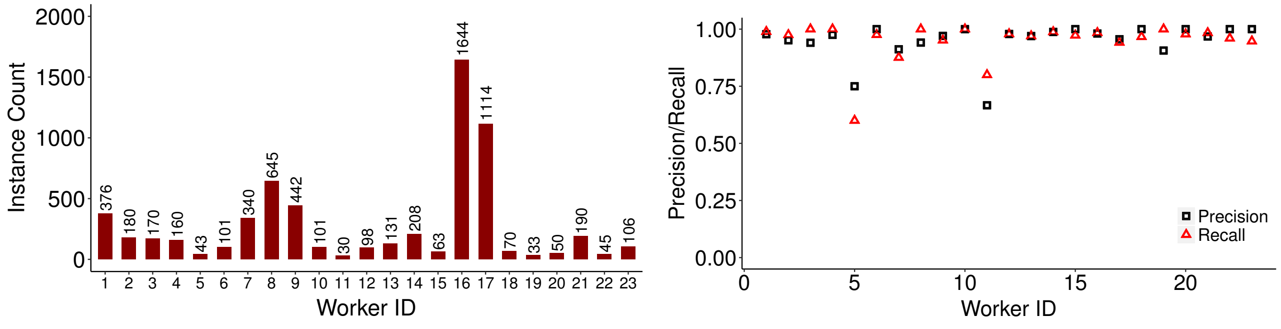
Developer tailored search rank fraud. Upon closer inspection of the DOLOS identified clusters, we found numerous cases of clusters consisting of user accounts who reviewed almost exclusively apps created by a single developer. We conjecture that those user accounts were created with the specific goal to review the apps of the developer, e.g., by the developer or their employees.

6.3 DOLOS in the Wild

To understand how Dolos will perform in real life, we have randomly selected 13,087 apps from Google Play, developed by 9,430 distinct developers. We monitored these apps over more than 6 months, and recorded their changes once every 2 days. This enabled us to collect up to 7,688 reviews per app, exceeding Google's one shot limit of 4,000 reviews. We collected the data of the 586,381 distinct reviewers of these apps, and built their co-activity graphs.

MCDense found at least 1 dense component of at least 5 accounts in 1,056 of the 13,087 apps (8%). Figure 10 compares the results of MCDense on the 1,056 apps, with

[3] https://en.wikipedia.org/wiki/Cross-validation_(statistics)



(a)

(b)

Fig. 11. (a) Number of review instances collected from each of the 23 fraud worker. Each review instance has at least 5 reviews, written by the accounts controlled by a single worker, for a single app. (b) DOLOS per-worker attribution precision and recall, over the 1,690 review instances of 640 fraud apps, exceed 87% for 21 out of the 23 workers.

those for the fraud and honest apps. The CDF of the number of components found by MCDense for these "wild" apps is closer to that of the fraud apps than to the honest apps: up to 19 components per app, see Figure 10(a). The CDF of the maximum density of per app components reveals that 231 of the 1,056 apps (or 21.87%) had at least 1 component with edge density 1 (complete sub-graphs). The CDF of the size of the densest components (Figure 10(c)) found per each of the wild apps shows that similar to the 640 fraud apps, few of these apps have only 0 size densest components. The largest component found by MCDense for these apps has 90 accounts.

Validation of fraud suspicions. Upon close inspection of the 231 apps that had at least 1 component with edge density of 1 (i.e., clique), we found the following further evidence of suspicious fraud being perpetrated. (1) **Targeted by known fraud workers**: 169 of the 231 apps had received reviews from the 23 known workers (§ 4.3). One app had received reviews from 10 of the workers. (2) **Review duplicates**: 223 out of the 231 apps have received 10,563 *duplicate* reviews (that replicate the text of reviews posted for the same app, from a different account), or 25.55% of their total 41,339 reviews. One app alone has 1,274 duplicate reviews, out of a total of 4,251 reviews. (3) **Fraud re-posters**: our longitudinal monitoring of apps enabled us to detect fraud re-posters, accounts who re-post their reviews, hours to days after Google Play filters them out. One of the 231 apps received 37 fraud re-posts, from the same user account.

6.4 MCDense Evaluation

6.4.1 MCDense Competitors

We adapt two existing solutions to the fraud-component detection problem and compare them against MCDense.

DSG: Adapted Densest SubGraph approach. We first compare MCDense against DSG, a densest subgraph approach that we adapt based on [25]. DSG, whose pseudocode is shown in Algorithm 3, iteratively identifies multiple dense subgraphs of an app's co-activity graph $G = (U, E)$, each suspected to belong to a different worker. DSG peels off nodes of G until it runs out of nodes (lines 4-11). During each "peeling" step, it removes the node that is least connected to the other nodes (lines 5-6). After removing the node, the algorithm computes and saves the density of the resulting subgraph (lines 7-10). The algorithm returns the subgraph with the highest density. We use the *triangle density* definition proposed in [25], $\rho_D = \frac{t(U)}{|U|}$, where $t(U)$ is the number of triangles formed by the vertices in U. DSG uses this greedy strategy iteratively: once it finds the densest subgraph D of G, DSG repeats the process, to find the densest subgraph in $G - D$. The nodes in each identified densest subgraph are well connected among themselves, but not well connected to the nodes in the previously identified subgraphs.

Algorithm 3 DSG: Densest Sub-Graph algorithm.

```
Input: G = (V, E): input graph
       n := $|V|$
Output: SG: optimum subgraph
1.  Graph H := G;
2.  double r := $\rho_D$(H); # holds max density
3.  Graph SG := G
4.  for i := 2 to n do
5.      v := least connected node of $H$;
6.      H := H - {v}
7.      if ($\rho_D$(H) $>$ r) then
8.          SG := H;
9.          r := $\rho_D$(H);
10.     end if
11. end for
12. return SG;
```

LBP: Adapted Loopy Belief Propagation approach. We adapt a Loopy Belief Propagation approach [18] to formulate the problem of detecting fraudulent user accounts as a network classification task on $\mathcal{G}$. The resulting algorithm, LBP, assigns labels to the user account nodes of the co-activity graph. Specifically, the graph is modeled as a pairwise Markov Random Field (MRF) [33], where each user account node has a random variable Y_i that can take values from the user class domain $\mathcal{L} = \{honest, fraud\}$ (i.e., the label space), encoding the belief that the node is fraudulent.

In MRFs, the memoryless Markov property implies that in the undirected network, the label of a node only depends on its neighbors. Then, the overall joint probability distribution is written as the normalized product of factors associated with the nodes and edges [34]: $\mathcal{P}(\mathbf{y}) = \frac{1}{Z} \prod_{Y_i \in \mathcal{U}} \phi_i(y_i) \prod_{(Y_i, Y_j, w_{ij}) \in \mathcal{E}_w} \psi_{ij}^w(y_i, y_j)$, where $\mathbf{y}$ represents an assignment of labels to each of the nodes in $\mathcal{U}$, and Z is the normalization constant. $\phi : \mathcal{L} \rightarrow \mathbb{R}^+$ represents the "prior probability" for each node, and $\psi^w : \mathcal{L} \times \mathcal{L} \rightarrow \mathbb{R}^+$ are the "compatibility potentials".

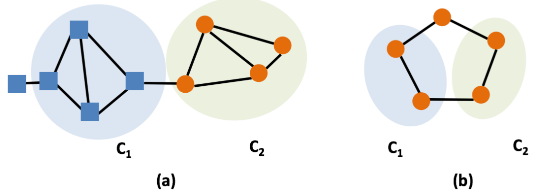


Fig. 12. Illustration of p-coverage and p-SCC scores that measure the quality of detected community partitions (shown as ovals). (a) **Graph with two workers**: one controls the square nodes and one the round nodes. The partition provides (50%,100%)-coverage and (50%,100%)-SCC. (b) **Graph with a single worker** (circles). p-coverage ≠ p-SCC: the partition provides (100%,80%)-coverage but only (0%,80%)-SCC.

We adapt the priors ϕ_i and compatibility potentials ψ_{ij} to capture the behavioral dynamics of the users in the Google Play review ecosystem. We have experimented with two types of priors. First, the priors are all 1/2, modeling the lack of knowledge on the status of nodes/accounts. Second, we chose the prior of a user node u_i having label *honest* to be inversely proportional to the average weight of u_i's edges, i.e., $\phi_i(honest) = \frac{|N(u_i)|}{\sum_{j \in N(u_i)} w_{ij}}$; $\phi_i(fraud) = 1 - \phi_i(honest)$. The results shown are over the latter approach, which proved to be more effective.

Further, the compatibility potentials $\psi_{ij}(y_i, y_j)$, capture the likelihood of a node u_i with assigned label y_i to be neighbor of a node u_j with label y_j, when u_i and u_j have a link of weight w between them. We defined $\psi_{ij}(y_i, y_j)$ as follows. If u_i is honest, $\psi_{ij}(y_i, y_j)$ is independent of w_{ij}. However, when u_i is "fraud" $\psi_{ij}(y_i, y_j)$ depends on w_{ij}: $\psi_{ij}(y_i = \text{“}fraud\text{''}, y_j = \text{“}honest\text{''}) = \delta^{\log w_{ij}}$ decreases exponentially with w_{ij}, while $\psi_{ij}(y_i = \text{“}fraud\text{''}, y_j = \text{“}fraud\text{''}) = 1 - \delta^{\log w_{ij}}$ increases, where δ in $(0, 1)$.

6.4.2 Evaluation Scores

To evaluate MCDense, we introduce two coverage scores. Let $\mathcal{RA}(A) = \{a_1, .., a_k\}$ denote the set of user accounts who reviewed product A. Given the set $S = \{W_1, .., W_w\}$ of workers who wrote fake reviews for A, let $\overline{W_i} = \{a_{i_1}, .., a_{i_j}\}$ denote the accounts in $\mathcal{RA}(A)$ that are controlled by worker W_i, $i = [w]$. Then, a *partition* of $\mathcal{RA}(A)$ is a set of sets $\{P_1, .., P_p\}$, such that each account $a_i \in \mathcal{RA}(A)$, $i = [k]$, belongs to exactly one of these sets. Let $H = \mathcal{RA}(A) \setminus \cup_{i=[w]} \overline{W_i}$, be A's "honest" reviewers, i.e., accounts not known to be controlled by a worker. The set $\{\overline{W_1}, .., \overline{W_w}, H\}$, forms a partition of $\mathcal{RA}(A)$.

Let $\mathcal{C} = \{C_1, .., C_c, H_C\}$ be the partition of $\mathcal{RA}(A)$ of the user accounts who reviewed an app A, returned by a fraud-component detection algorithm: $\forall a_i, a_j \in C_l$, are considered to be controlled by the same worker, and H_C is the set of accounts considered to be honest. To quantify how well the partition $\mathcal{C}$ has detected the worker accounts $\overline{W_1}, .., \overline{W_w}$ who targeted A, we propose the *coverage* measure of worker $W_i \in S$ as follows:

Definition 1. (Coverage) The *coverage* of worker $W_i \in S$ by a partition C is $cov_i(C) = \frac{|\overline{W_i} \cap (C_1 \cup .. \cup C_c)|}{|\overline{W_i}|}$. Given $p \in [0, 1]$, we say that W_i is "p-covered" by C if $cov_i(C) \geq p$. Then, we say that partition C provides a (p_1, p_2)-coverage of the worker set S, if p_1 percent of the workers in S are p_2-covered by C.

Further, we introduce the *single component coverage* (SCC) of a worker $W_i \in S$ by a partition $\mathcal{C}$:

Definition 2. (Single Component Coverage - SCC) The *single component coverage* of a worker $W_i \in S$ by a partition $\mathcal{C} = \{C_1, .., C_c\}$ is $\text{SCC}_i(C) = max_{j=[c]} \frac{|\overline{W_i} \cap C_j|}{|\overline{W_i}|}$. Given $p \in [0, 1]$, we say that W_i is "p-single component covered" (or p-SCC) by C if $\text{SCC}_i(C) \geq p$. We say that partition C provides a (p_1, p_2)-SCC of the worker set S, if p_1 percent of the workers in S are p_2-SCC by C.

p-SCC is about precision: a worker is p-single component covered by Alg only if at least p percent of its accounts belong to a single component discovered by Alg. In contrast, a worker is p-covered if p percent of its accounts belong to any component returned by Alg. As such, p-SCC will always be at most equal to p-coverage. Figure 12 illustrates the p-coverage and p-SCC measures on the co-activity graphs of two apps.

6.4.3 MCDense vs. DSG

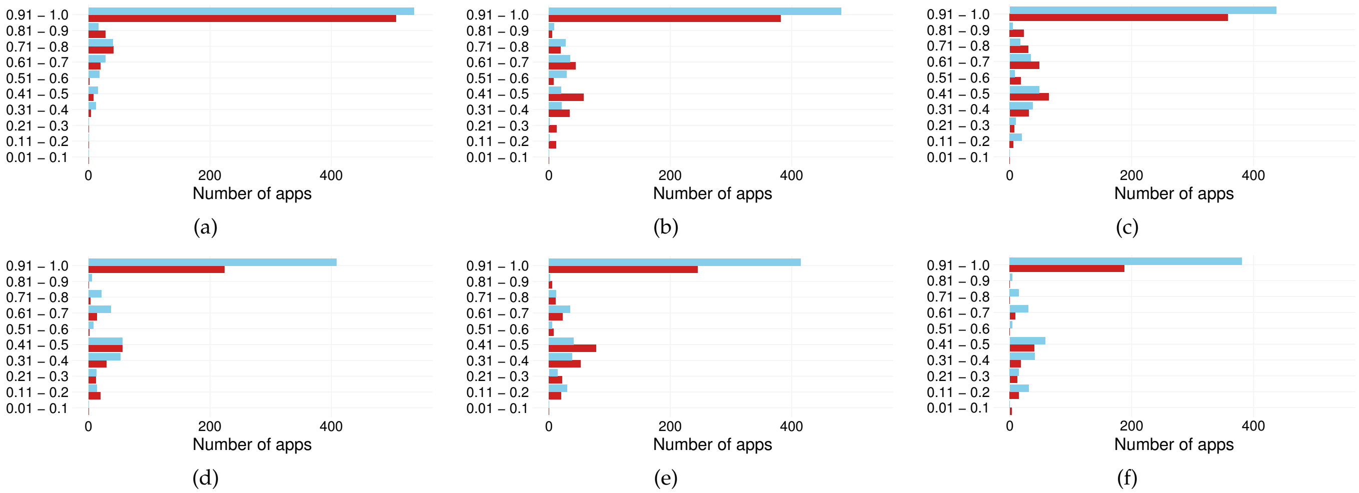
Figure 13 compares MCDense and DSG in terms of their distributions of the p-coverage and p-SCC scores, over the 640 fraud apps. MCDense consistently outperforms DSG. For instance, Figure 13(a) shows that 537 apps are at least (90%+, 50%)-covered by MCDense, while only 507 apps achieve the same coverage for DSG. The difference is even higher for the p-SCC score: Figure 13(b) shows that 490 apps (75%) are at least (90%+, 50%)-SCC by MCDense, that is, at least 50% of the accounts controlled by 90% of the workers belong to *only one* of the components returned by MCDense. In contrast, only 383 apps are at least (90%+, 50%)-SCC by DSG.

The difference between MCDense and DSG becomes more pronounced as p_2 increases to 80% and 90%. For instance, Figure 13(c) shows that 438 apps are at least (90%+, 80%)-covered by MCDense, while only 359 apps achieve the same coverage for DSG. Figure 13(d) compares the p-SCC score: 409 apps (63%) are at least (90%+, 80%)-SCC by MCDense compared to only 225 apps that are at least (90%+, 80%)-SCC by DSG. Figure 13(e) shows that 415 apps are at least (90%+, 90%)-covered by MCDense, while only 245 apps achieve the same coverage for DSG. Figure 13(f) shows the p-SCC score: 381 apps (59%) are at least (90%+, 90%)-SCC by MCDense, but only less than half (188 apps) are at least (90%+, 90%)-SCC by DSG.

6.4.4 LBP Performance

Since LBP has no information (in the form of priors) about the accounts controlled by workers, we use it to determine which accounts are suspected of being controlled by *a* worker, as those whose final fraud belief exceeds 0.5. Figure 14 shows the box and whiskers plot of the precision and recall values of LBP, when δ ranges from 0.1 to 0.9. We observe that recall in this case is equivalent to our p-coverage score. In addition to precision and recall, we also use the notion of "prevalence": the ratio of the number of fraud labeled accounts to the total number of the app's accounts. This enables us to determine when LBP labels all the accounts as fraudulent. LBP achieves the best performance when δ = 0.7, with an average per-app recall exceeding 95%.

Thus, while LBP can be used to detect fake reviews, it cannot determine if all accounts detected as fraudulent are controlled by a single or multiple workers.



(a) (b) (c)

(d) (e) (f)

Fig. 13. Comparison of MCDense and DSG **distribution of coverage score** and **distribution of SCC score** over 640 fraud apps. The y axis shows the p_1 value, and the x axis shows the number of apps for which MCDense and DSG achieve that p_1 value, when (a,b) p_2 = 50%, (c,d) p_2 = 80%, and (e,f) p_2 = 90%. **MCDense consistently outperforms DSG** as it provides (a) (90%+, 50%)-coverage for 537 (83%) of the apps vs. DSG's 506 apps, and (b) (90%+, 50%)-SCC for 490 (75%) of the apps, vs DSG's only 383 apps, and (e) (90%+, 90%)-coverage for 415 (65%) of the apps vs. DSG's 245 apps, and (f) (90%+, 90%)-SCC for 381 of the apps, vs. DSG's 188 apps, which is half of MCDense.

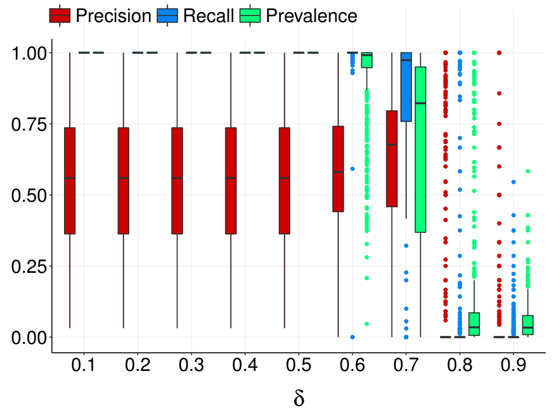


Fig. 14. LBP identification of fraudulent accounts: precision, recall and prevalence, when $\delta = [0.1..0.9]$. LBP achieves best performance when $\delta = 0.7$, with a median precision of 69% and recall of 98%; the median prevalence of 82% shows that these values are not achieved by labeling all the accounts as fraudulent.

7 Discussion and Limitations

Union fraud graph complexity. Given n unique user accounts, the complexity of building their union fraud graph is $O(n^2)$: the existence of an edge and its weight, need to be identified and computed for each pair of user accounts. The addition or deletion of a single account to/from the union fraud graph has $O(n)$ complexity, as the account may have at most $n - 1$ edges to the other nodes in the graph.

Unexpected fraud and generalization of results. Dolos is designed to work with fraud workers defined by the traits of § 4.5. Thus, Dolos may not be effective in identifying and attributing fraud posted by workers who do not share these traits.

Further, we performed our experiments using data we collected from Google Play. We conjecture that Dolos could also attribute fraud to workers who target other peer-opinion sites, e.g., Amazon, Yelp, App Store, as long as they satisfy the traits of § 4.5. However, due to lack of data, we cannot validate this hypothesis, and cannot claim that our results generalize to other sites.

Automating fraud discovery. Our approach requires manual interaction with fraud workers, in order to collect attributed fraud, i.e., user accounts that they claim to control. While it would be desirable to fully automate fraud attribution efforts, we note that informed consent is required to interact with and collect data from human workers. Commercial peer-opinion sites may use alternative techniques to automatically attribute accounts to workers, e.g., through duplicate IP addresses used during site accesses. However, fraud workers can also use VPNs to de-anonymize their access. We leave an investigation into ethical and automatic collection of attributed fraud for future work.

Answer validation. Our studies involving fraud workers assume that their answers are truthful. While we verified that workers knew the Gmail accounts associated with the accounts that they claimed to control, we have less confidence in answers to questions about, e.g., the number of reviews they can write, fraud experience and number of apps reviewed in the past week (shown in Figure 3). Fraud workers may have incentives to exaggerate expertise. Future work may investigate protocols to validate answers provided by fraud workers.

8 Related Work

This article significantly extends our initial work [35] with (1) a detailed analysis of fraud data that we collected from 23 crowdsourced workers, an extension of the results of our study on search rank fraud jobs, and additions to the extracted adversarial traits, (2) the addition of p-SCC, a new coverage score, (3) pseudocode for Dolos and DSG, (4) significant new comparison of MCDense and DSG, both for the previous p-coverage and the new p-SCC score, and (5) the introduction of a Loopy Belief Propagation algorithm adapted to the fraud detection problem, and its evaluation.

Fraud detection in peer-review sites. Yang et al. [36] showed that "criminal" Twitter accounts tend to form small-world social networks. Mukherjee et al. [11], [14] confirmed this finding and introduced features that identify reviewer groups, who review many products in common but not much else, post their reviews within small time windows, and are among the first to review the product.

Beutel et al. [37] proposed CopyCatch, a system that identifies *lockstep behaviors*, i.e., groups of user accounts that act in a quasi-synchronized manner, to detect fake page likes in Facebook. Chen et al. [9] identify clusters of apps in Apple's China App store, that have been promoted in a similar fashion. Cao et al. [38] used lockstep behaviors and insider information, to cluster user accounts and uncover groups of malicious accounts in Facebook.

DOLOS goes beyond fraud detection, to further attribute detected fraud to the human fraud workers in crowdsourcing sites who are responsible for posting it.

Graph-based fraud detection. Previous work has used graph based approaches to detect fraudulent behaviors, e.g., [39], [12], [40], [19]. Ye and Akoglu [39] quantified the chance of a product to be a spam campaign target, then clustered spammers on a 2-hop subgraph induced by the products with the highest chance values. Wang et. al [12] leveraged a novel Markov Random Field to detect workers in social networks via guilt-by-association on directed graphs. Shen et al [40] introduced "k-triangles" to measure the tenuity of account groups and proposed algorithms to approximate the Minimum k-Triangle Disconnected Group problem. Hooi et al. [19] have shown that workers have evolved to hide their traces, by adding spurious reviews to popular items. They introduced a class of "suspiciousness" metrics that apply to bipartite user-to-item graphs, and developed a greedy algorithm to find the subgraph with the highest suspiciousness metric.

Wang et al. [41] used "heterogeneous review graphs" that capture relations among reviewers, reviews and subjects, and develop an iterative model to identify suspicious reviewers. Malliaros et al. [42] exploit *expansion properties* of large social graphs to build an efficient algorithm for computing the robustness property of time evolving graph to detect communities and anomalies. Faloutsos et al. [43] use static and temporal properties (e.g. eigenspokes) of time-evolving graphs to spot suspicious activities and process time-evolving graphs in map-reduce environments.

DOLOS builds on the empirical observation that search rank fraud is perpetrated by many workers who often control hundreds of user accounts. To de-anonymize the influential workers, DOLOS leverages co-activity graphs that capture the frequency and intensity of common activities posted from user accounts.

Stylometry based fraud detection and de-anonymization. Ott et al. [29] used computational linguistics features to detect deceptive TripAdvisor reviews. Lau et al. [30] proposed a text mining model integrated into a semantic language model to detect fake Amazon reviews. Sentiment and bias, e.g., [44] may complement stylometry tools to help attribute detected fraud. Overdorf and Greenstadt [31] proposed authorship attribution methods that work across social networks. Abbasi and Chen [45] developed Writeprints, a system for de-anonymizing e-mail, IM, reviews and program code. Narayanan et al. [46] proposed author de-anonymization techniques that handle large number of classes ($100,000$ authors).

DOLOS links search rank fraud to crowdsourcing site worker accounts, thus breaks the anonymity barrier between crowdsourcing site workers and Google Play user accounts. DOLOS is resilient to input imprecision (due to the false positives of its fraud component detection module).

9 CONCLUSIONS

We introduced the fraud de-anonymization problem for search rank fraud in online services. We have collected fraud data from crowdsourcing sites and the Google Play store, and we have performed user studies with crowdsourcing workers. We have proposed DOLOS, a fraud de-anonymization system. DOLOS correctly attributed 95% of the fraud detected for 640 Google Play apps, and identified at least 90% of the workers who promoted each of 87% of these apps. DOLOS identified 1,056 out of 13,087 monitored Google Play apps, to have suspicious reviewer groups, and revealed a suite of observed fraud behaviors. DOLOS significantly outperforms adapted dense subgraph detection and loopy belief propagation competitors, in two coverage scores that we have developed.

10 ACKNOWLEDGMENTS

This research was supported by NSF grants CNS-1527153 and CNS-1526254 and by the Florida FC^2 Center.

REFERENCES

[1] Huffington Post. Yelp study shows extra half-star nets restaurants 19reservations. Huffington Post, https://tinyurl.com/y7u32ssl, 2012.

[2] Michael Luca. Reviews, Reputation, and Revenue: The Case of Yelp.Com. *SSRN eLibrary*, 2011.

[3] Google I/O 2013 - Getting Discovered on Google Play. www.youtube.com/watch?v=5Od2SuL2igA, 2013.

[4] Fiverr. https://www.fiverr.com/.

[5] Upwork Inc. https://www.upwork.com.

[6] Freelancer. http://www.freelancer.com.

[7] Zeerk. https://zeerk.com/.

[8] Peopleperhour. https://www.peopleperhour.com/.

[9] Hao Chen, Daojing He, Sencun Zhu, and Jingshun Yang. Toward detecting collusive ranking manipulation attackers in mobile app markets. In *Proceedings AsiaCCS*, 2017.

[10] Mahmudur Rahman, Mizanur Rahman, Bogdan Carbunar, and Polo Chau. Fairplay: Fraud and Malware Detection in Google Play. In *Proceedings of the SIAM International Conference on Data Mining (SDM)*, 2016.

[11] Arjun Mukherjee, Bing Liu, and Natalie Glance. Spotting fake reviewer groups in consumer reviews. In *Proceedings of ACM WWW*, 2012.

[12] Binghui Wang, Neil Zhenqiang Gong, and Hao Fu. GANG: Detecting Fraudulent Users in Online Social Networks via Guilt-by-Association on Directed Graphs. In *Proceedings of ICDM*, 2017.

[13] Geli Fei, Arjun Mukherjee, Bing Liu, Meichun Hsu, Malu Castellanos, and Riddhiman Ghosh. Exploiting burstiness in reviews for review spammer detection. *ICWSM*, 13:175–184, 2013.

[14] Arjun Mukherjee, Abhinav Kumar, Bing Liu, Junhui Wang, Meichun Hsu, Malu Castellanos, and Riddhiman Ghosh. Spotting opinion spammers using behavioral footprints. In *Proceedings of the ACM KDD*, 2013.

[15] Huayi Li, Geli Fei, Shuai Wang, Bing Liu, Weixiang Shao, Arjun Mukherjee, and Jidong Shao. Bimodal distribution and co-bursting in review spam detection. In *Proceedings of ACM WWW*, 2017.

[16] Bryan Hooi, Neil Shah, Alex Beutel, Stephan Günnemann, Leman Akoglu, Mohit Kumar, Disha Makhija, and Christos Faloutsos. Birdnest: Bayesian inference for ratings-fraud detection. In *Proceedings of SDM*, 2016.

[17] Prudhvi Ratna Badri Satya, Kyumin Lee, Dongwon Lee, Thanh Tran, and Jason Jiasheng Zhang. Uncovering fake likers in online social networks. In *Proceedings of the ACM CIKM*, 2016.

[18] Leman Akoglu, Rishi Chandy, and Christos Faloutsos. Opinion Fraud Detection in Online Reviews by Network Effects. In *Proceedings of ICWSM*, 2013.

[19] Bryan Hooi, Hyun Ah Song, Alex Beutel, Neil Shah, Kijung Shin, and Christos Faloutsos. Fraudar: Bounding graph fraud in the face of camouflage. In *Proceedings of ACM KDD*, 2016.

[20] Shebuti Rayana and Leman Akoglu. Collective opinion spam detection: Bridging review networks and metadata. In *Proceedings of ACM KDD*, 2015.

[21] Leman Akoglu, Rishi Chandy, and Christos Faloutsos. Opinion fraud detection in online reviews by network effects. *Proceedings of ICWSM*, 2013.

[22] Jason Cipriani. Google starts filtering fraudulent app reviews from Play Store. ZDNet, https://tinyurl.com/hklb5tk, 2016.

[23] Sarah Perez. Amazon bans incentivized reviews tied to free or discounted products. Tech Crunch, https://tinyurl.com/zgn9sq3, 2016.

[24] Arjun Mukherjee, Vivek Venkataraman, Bing Liu, and Natalie Glance. What Yelp Fake Review Filter Might Be Doing. In *Proceedings of ICWSM*, 2013.

[25] Charalampos E. Tsourakakis. The k-clique densest subgraph problem. In *Proceedings of ACM WWW*, 2015.

[26] Dolos on github. https://github.com/FraudHunt.

[27] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of ACM KDD*, 2014.

[28] David R Karger. Global min-cuts in rnc, and other ramifications of a simple min-cut algorithm. In *SODA*, volume 93, 1993.

[29] Myle Ott, Yejin Choi, Claire Cardie, and Jeffrey T. Hancock. Finding deceptive opinion spam by any stretch of the imagination. In *Proceedings of the Human Language Technologies*, HLT '11, 2011.

[30] Raymond YK Lau, SY Liao, Ron Chi Wai Kwok, Kaiquan Xu, Yunqing Xia, and Yuefeng Li. Text mining and probabilistic language modeling for online review spam detecting. *ACM Transactions on Management Information Systems*, 2(4):1–30, 2011.

[31] Rebekah Overdorf and Rachel Greenstadt. Blogs, twitter feeds, and reddit comments: Cross-domain authorship attribution. *PoPETs*, 2016(3), 2016.

[32] JStylo. The JStylo Open Source Project on Open Hub. https://www.openhub.net/p/jstylo.

[33] Jonathan S Yedidia, William T Freeman, and Yair Weiss. Understanding belief propagation and its generalizations. *Exploring artificial intelligence in the new millennium*, 8, 2003.

[34] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models. Principles and Techniques.* The MIT Press, 2009.

[35] Mizanur Rahman, Nestor Hernandez, Bogdan Carbunar, and Duen Horng Chau. Search rank fraud de-anonymization in online systems. In *Proceedings of the ACM Conference on Hypertext and Social Media*, 2018.

[36] Chao Yang, Robert Harkreader, Jialong Zhang, Seungwon Shin, and Guofei Gu. Analyzing spammers' social networks for fun and profit: a case study of cyber criminal ecosystem on Twitter. In *Proceedings of ACM WWW*, 2012.

[37] Alex Beutel, Wanhong Xu, Venkatesan Guruswami, Christopher Palow, and Christos Faloutsos. CopyCatch: Stopping Group Attacks by Spotting Lockstep Behavior in Social Networks. In *Proceedings of the WWW*, 2013.

[38] Qiang Cao, Xiaowei Yang, Jieqi Yu, and Christopher Palow. Uncovering large groups of active malicious accounts in online social networks. In *Proceedings of ACM CCS*, 2014.

[39] Junting Ye and Leman Akoglu. Discovering opinion spammer groups by network footprints. In *Machine Learning and Knowledge Discovery in Databases*. 2015.

[40] Chih-Ya Shen, Liang-Hao Huang, De-Nian Yang, Hong-Han Shuai, Wang-Chien Lee, and Ming-Syan Chen. On finding socially tenuous groups for online social networks. In *Proceedings of KDD*, 2017.

[41] Guan Wang, Sihong Xie, Bing Liu, and Philip S. Yu. Review Graph Based Online Store Review Spammer Detection. *IEEE ICDM*, 2011.

[42] Fragkiskos D Malliaros, Vasileios Megalooikonomou, and Christos Faloutsos. Fast robustness estimation in large social graphs: Communities and anomaly detection. In *Proceedings of the 2012 SIAM International Conference on Data Mining*, pages 942–953. SIAM, 2012.

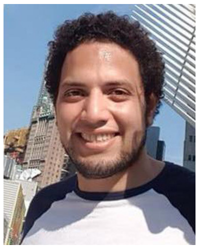
[43] Christos Faloutsos. Large graph mining: patterns, cascades, fraud detection, and algorithms. In *Proceedings of the 23rd international conference on World wide web*, pages 1–2. ACM, 2014.

[44] Haokai Lu, James Caverlee, and Wei Niu. Biaswatch: A lightweight system for discovering and tracking topic-sensitive opinion bias in social media. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 213–222. ACM, 2015.

[45] Ahmed Abbasi and Hsinchun Chen. Writeprints: A stylometric approach to identity-level identification and similarity detection in cyberspace. *ACM Transactions on Information Systems (TOIS)*, 26(2):7, 2008.

[46] Arvind Narayanan, Hristo Paskov, Neil Zhenqiang Gong, John Bethencourt, Emil Stefanov, Eui Chul Richard Shin, and Dawn Song. On the feasibility of internet-scale author identification. In *2012 IEEE Symposium on Security and Privacy*, 2012.

Mizanur Rahman is a Ph.D. candidate at Florida International University. He has previously held various positions in KAZ Software, iAppDragon and Prolog Inc. His research interest include internet data privacy, fraud detection in social network and user experience analysis.



Nestor Hernandez is a PhD. candidate at Florida International University. He has worked as a data mining analyst for Wunderman. His research interests include fraud and opinion spam detection in online services.

Duen Horng (Polo) Chau is an associate professor at Georgia Techs School of Computational Science and Engineering, and an Associate Director of the MS Analytics program. Polo holds a Ph.D. in Machine Learning and a Masters in human-computer interaction (HCI). Polo received faculty awards from Google, Yahoo, LexisNexis, and the Raytheon Faculty Fellowship and Edenfield Faculty Fellowship, Outstanding Junior Faculty Award.

Bogdan Carbunar is an associate professor in SCIS at FIU. Previously, he held various researcher positions within the Applied Research Center at Motorola. His research interests include distributed systems, security and applied cryptography. He holds a Ph.D. in Computer Science from Purdue University.