Mouawi et al. EURASIP Journal on Information Security (2019) 2019:11
https://doi.org/10.1186/s13635-019-0095-1

EURASIP Journal on Information Security

RESEARCH Open Access

Crowdsourcing for click fraud detection

Riwa Mouawi*, Imad H. Elhajj, Ali Chehab and Ayman Kayssi

Abstract

Mobile ads are plagued with fraudulent clicks which is a major challenge for the advertising community. Although popular ad networks use many techniques to detect click fraud, they do not protect the client from possible collusion between publishers and ad networks. In addition, ad networks are not able to monitor the user's activity for click fraud detection once they are redirected to the advertising site after clicking the ad. We propose a new crowdsource-based system called Click Fraud Crowdsourcing (CFC) that collaborates with both advertisers and ad networks in order to protect both parties from any possible click fraudulent acts. The system benefits from both a global view, where it gathers multiple ad requests corresponding to different ad network-publisher-advertiser combinations, and a local view, where it is able to track the users' engagement in each advertising website. The results demonstrated that our approach offers a lower false positive rate (0.1) when detecting click fraud as opposed to proposed solutions in the literature, while maintaining a high true positive rate (0.9). Furthermore, we propose a new mobile ad charging model that benefits from our system to charge advertisers based on the duration spent in the advertiser's website.

Keywords: Click fraud, Crowdsource, In-app ads, CPA, Mobile charging model, Android

1 Introduction

With the increasing number of free apps in the app stores (as high as 95.2% of the total available apps in Google Play Store according to a recent study [1]), in-app ads are gaining more popularity among developers who wish to generate revenues from their free apps. In fact, according to statistics published by Statistica, it is predicted for worldwide mobile app store revenues in 2020 to reach around $71.7 billion [2].

The term "in-app ads" represents ads that are displayed in mobile applications, whereas the term "mobile ads" actually represents ads that are displayed on smartphones in general (in both applications and mobile websites). Similarly to the literature, we will be using these terms interchangeably.

There are four main components in the mobile advertising community (Fig. 1):

1) Advertisers: individuals (not necessarily technical experts) or companies who are willing to pay to have their ads displayed.
2) Publishers: app developers who wish to generate revenues by displaying ads in their applications.
3) Ad network: a company specialized in mobile advertising, which works as a relay between advertisers and publishers. Advertisers contact the ad network and specify the ads to be displayed. Publishers also contact the ad network and follow an integration process (explained in Section 3).
4) Users: individuals that use a mobile application and interact with ads featured in their application (interaction types are explained below).

1.1 Mobile ad click fraud detection

Mobile ad click fraud is a major concern in the advertising community. Click fraud, also known as click spam, is when the user clicks on an ad in a mobile application not because of interest in the ad, but rather to generate a revenue from the associated ad network or, in some cases, to inflict losses on a competitor advertiser by consuming the advertiser's allowed ads per day [3]. Researchers estimate advertisers' loss caused by click fraud at $1 billion in 2013 [4] and at $7.2 billion in 2016 [47].

Furthermore, in [5], the author describes a $7.5 billion scandal in the click fraud domain, where ad networks conspired with publishers, by selling bots to publishers who used them in order to generate higher revenues (for both publisher and ad network) at the expense of the advertiser.

* Correspondence: riwamouawi@gmail.com
Department of Electrical and Computer Engineering, American University of Beirut, Beirut 1107 2020, Lebanon

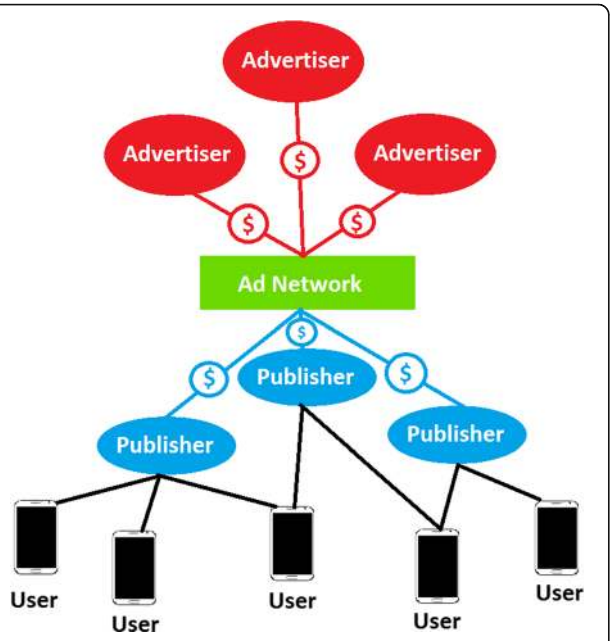

Fig. 1 Mobile advertising main components

Although most of the ad networks use many click fraud detection techniques to protect their reputation as a secure advertising medium, they do not offer guarantees to the client from a potential conspiracy between publishers and ad networks. In addition, ad networks are not able to monitor the users' activities once redirected to the advertising site. Thus, a click fraud detection mechanism that protects the advertiser from potential malicious ad network–publisher collaboration is needed.

Many researchers investigated existing click fraud attacks [6–12] and proposed new click fraud detection systems for ad networks without collaborating with advertisers [13–17]. Other studies proposed systems that enable advertisers to detect click fraud without collaborating with ad networks [18–20]. Nonetheless, these advertisers' click fraud systems are prone to a high false positive rate since they do not offer a global view (many clicks from one application), but instead they judge click fraud on a per click basis.

In the first part of this paper, we propose a new click fraud detection system that adapts a crowdsource-based approach to detect click spam by collaborating with the different advertisers that wish to display their ads in a secure and reliable way. In fact, our findings show that a crowdsource approach can lower the false positive rate. We make two main contributions in the click fraud detection domain since our work is (1) the first crowdsource-based click fraud detection system and (2) the first approach that protects both advertisers and ad networks. In the second part of the paper, we propose a new mobile ad charging model that benefits from our CFC system to charge advertisers based on the duration spent in the advertiser's website. Our proposed model in this paper is an enhancement of our previous work [48] where we briefly introduced our model and compared multiple classification methods; our model since then has matured and is following a different click fraud detection analysis method.

1.2 Mobile ad charging models

There are currently three main mobile ad charging models used in the market: (1) cost per thousand impressions (CPM from cost per mille in Latin), (2) cost per click (CPC), and (3) cost per action (CPA). These charging models determine how for each ad display and interaction, the ad network charges the advertisers and how it pays the publishers. Although the exact pricing differs among ad networks (i.e., how much they pay the publisher and charge the advertiser), the action that triggers this transaction depends on the adopted charging model.

To better understand each charging model, we need first to explain the term "conversion," which is used in the mobile advertising community to express a successful advertising transaction, i.e., whenever an ad display leads to the desired output (the type of the output depends on the charging model) [21].

In the cost-per-thousand impressions (CPM) charging model, the conversion is an ad view: The advertiser pays for each 1000 ad displays; the publisher is paid in return a percentage of the ad network's profit whenever a user views an ad in their application.

In the cost-per-click (CPC) charging model, the conversion is an ad click: The advertiser pays for each performed ad click; the publisher is paid in return whenever a user clicks on a displayed ad.

In the cost-per-action (CPA) model, the conversion is an action: The advertiser pays for each performed action. Once the user clicks on a mobile ad, they will be redirected to the advertiser's website where they might complete an action. An action can be a simple registration by the user, a purchase, a file download, etc. The publisher is paid in return for each action performed. The CPA model is also known as the pay-per-action (PPA) model or cost-per-acquisition (CPA) model.

The CPC and CPM models are usually more desirable by publishers [22] for many reasons:

1) CPA model is considered a new model in the mobile advertising community and is not as popular as CPC and CPM.
2) In CPC and CPM models, publishers have more control on the enhancement of conversions, since they can change how and when to display ads (to a certain extent because they have to respect ad-display rules imposed by ad networks). For example, they can

display ads on main pages usually accessed by users, whereas, in the CPA model, the conversion is beyond their control and is based on the quality of the advertiser's website or service.

3) Many users, after being redirected to the advertiser's website, show interest in the website by browsing it for a certain duration. However, it is possible that although they are interested in the website, they will not complete any of the actions determined by current CPA models. In fact, in [21], the authors explained a "click to conversion delay" phenomenon in the mobile advertising industry, where the action performed by the user is not directly performed after clicking on an ad, making it harder to track CPA conversions. For example, a user interested in cosmetic products clicks on a cosmetic ad and browses the advertiser's website without completing any purchases. The next week, after they had become aware of this website, they browse the cosmetics website again directly and make a purchase. In this case, it is hard to accurately associate the conversion to the triggering event. Furthermore, many websites' main goal is brand awareness where no action is needed by the user. Thus, CPA models require a metric that efficiently reflects the user's interest in the advertised website such as the duration spent on the website.

On the other hand, advertisers tend to prefer the CPA model because they pay only for the desired actions once completed. In the CPC and CPM models, the advertisers pay for the ad views and clicks regardless of whether it generates users interests (conversions in this context), making the CPA model less risky for them.

We propose a new mobile ad charging model that benefits from our CFC system to charge advertisers based on the duration spent in the advertiser's website.

This new mobile ad charging model is transparent to advertisers, since it does not require any integration steps to be performed from their part on their website. It also presents a secure framework in which the action to be charged for (such as the duration) is measured by a trusted party as opposed to current CPA systems that rely on the advertisers to report the action.

The rest of this paper is organized as follows: in Section 2, we discuss the related work highlighting the proposed click fraud detection systems in the literature and the related work of the CPA domain. In Section 3, we present the architecture of our proposed system. In Section 4, we explain the experimental setup. In Sections 5 and 6, we present the results of our implementation and analyze the obtained results. Section 7 concludes the paper and offers suggestions for future work.

2 Related work

2.1 Click fraud detection

In recent studies, three approaches have been followed in terms of evaluating different click fraud techniques, impact, and solutions.

2.1.1 Click fraud investigation and analysis

The first approach investigates and measures the prevalence of existing click fraud attacks and threats. However, none of these studies proposed a tool that can be used to defend against click spam.

Dave et al. conducted a click fraud measurement analysis where they showed that for a certain ad network, over 95% of users redirected to their website after clicking on an ad, spent less than 1 s on their landing page [3]. Kshitri et al. presented an overview of click fraud from an economical perspective and explained different existing click fraud detection mechanisms [34]. Zhu et al. listed several available commercial tools used to detect and prevent ad fraud. They briefly described the approaches followed by these tools [40]. Berrar et al. evaluated several techniques in supervised learning by adopting click fraud as a case study [39]. Cho et al. created an automated click fraud tool that generates virtual ad clicks while changing the device identifier to a random value with each request and proved that six out of eight ad networks were actually vulnerable to this attack. Furthermore, they explained how ad networks might not actually be motivated to fight against click fraud since click fraud inflicts loss on the advertiser's side only, whereas ad networks might actually profit from the fraudulent traffic that is generated by malicious bots [6, 46]. Wen et al. created an emulator-based device capable of conducting a large-scale click fraud attack that ran successfully on over 280 applications downloaded from the Google Play Store [49]. Liang et al. studied the difference between click fraud generated by publishers and click fraud generated by ad networks [52]. Their study showed a correlation between a decrease of fraud committed by ad networks when facing long contractual advertising agreements with advertisers. Dmytro et al. evaluated the performance of an existing signaling processing known as "Caterpillar-SSA" on a real-life case study [58].

Following a machine learning approach while testing over 165 K apps for click fraud, Crussell et al. showed that 30% of apps requested ads while in the background and 27% of apps generated clicks without any user interaction [7]. Blizzard et al. studied an existing malware on ad websites that generates fraudulent clicks by redirecting the user to an intermediate page where the user has to click again on the ad (to double the revenue) [8]. Alrwais et al. conducted a detailed investigation of a large-scale cybercriminal attack known as "Operational Ghost Click," where attackers used a DNS changer malware to

hijack ad impressions and ad clicks from victim publishers [9]. Kayalvizhi et al. presented a survey on click fraud by comparing the advantages and disadvantages of several fraud detection tools [51].

Stone-Gross et al. presented a detailed analysis of how ad exchange (where ad networks sell/buy their publisher's ad space to/from another ad network) actually works and what are the different ad fraud threats in ad exchange [10]. Pearce et al. used real ad traffic traces provided by an ad network to study the behavior of a famous large-scale click fraud botnet called "ZeroAccess." Based on the analyzed behavior of this botnet, they estimated the loss of revenues from the advertiser's side to be around 100 K per day [11]. Miller et al. operated two families of bots in a controlled environment to monitor how botnet ad fraud, also known as clickbot, works in action [12].

Based on data collected from 155 subjects, Midha developed an ethical behavioral model that analyzes the reasons for which malicious users commit click fraud in web advertising [36]. Dinev et al. also conducted a study of advertiser's behavior towards the pay-per-click model in web advertising, based on data collected from 118 advertisers and highlighted the importance of hiring third party tools to manage the advertising model [38]. Zhu et al. presented a comprehensive review of ad fraud in general and explained the different ad fraud prevention and detection approaches and tools that are currently being used in the market [43]. Kshetri et al. explained how a blockchain-based solution might be a new solution for click fraud [55]. Dong et al. studied different ad fraud types and proposed a detection technique for fraud related to placement ads [57].

2.1.2 Click fraud detection—ad network's side

The second category in the literature focuses on click fraud detection on the ad network's side without collaborating with advertisers: Liu et al. created a tool that detects click fraud attacks known as "placement ads" such as hidden ads, ads out of context, and numerous ads per page [13]. Juels et al. proposed a new ad fraud mitigation technique that protects the pay-per-click charging model by cryptographically authenticating legitimate users [14]. Vasumati et al. used a data mining classification algorithm to identify fraudulent clicks [15]. Li et al. proposed a new ad framework that uses ARM Trustzone services to securely fetch ad placement and application-related information, and then signs this information with the user's signature [16]. Haddadi et al. proposed a new click fraud detection technique where they fabricated random ads, known as bluff ads, with the assumption that a non-malicious user would not normally click a random ad when its content is not relevant to them [17]. Asdemir et al. highlighted the benefits of adopting a prescreening approach in web advertising by obligating publishers to provide to search engines (ad networks) proof of good content such as third party audience reports and website visit analytics data [37]. Faou et al. conducted a study based on 7 months of data collected from a known click fraud malware, to identify the key actors of the fraud industry and proved that the malware can be seriously disrupted by removing a limited subset of the identified actors [35]. Zhang et al. presented their click fraud detection algorithm that detects coalition attacks in websites where a group of attackers share their resources to hide their trace while performing click fraud [41]. Using real-life data set from ad data management platforms in China, Jianyu et al. performed a supervised learning classification method to determine the key attributes that should be taken into consideration when detecting ad fraud such as IP address, cookie, and user-agent with their frequency [44]. Nagaraja et al. leveraged timing patterns of click traffic to identify fraudulent acts [54]. Haider et al. proposed an ensemble-based classifier that detects fraudulent acts in impression ads [59].

2.1.3 Click fraud detection—advertiser's side

The third category of literature addresses click fraud detection on the advertiser's side without collaborating with ad networks. Dave et al. proposed a new framework that enables advertisers to detect potential spam clicks on their ads based on several criteria such as the duration spent by the user on their websites [3]. Xu et al. proposed a new ad fraud detection mechanism deployed and managed by the advertisers. First, they identify bots by checking browser-related information of the user after clicking an ad and landing on the advertiser's page, and then they identify sophisticated bots or human clickers, by monitoring user behavior such as the duration spent on the advertiser's website and mouse events [18]. Vani et al. used network ad traces collected from the advertiser's side to extract click-related information (such as user IP address, user agent values, and time of access) and employed a node-tag-based algorithm to differentiate spam and non-spam applications [19].

Iqbal et al. created a tool integrated within the operating system's anti-virus that detects click fraud in websites by analyzing HTTP requests that are not associated to real human behavior such as mouse events. Although their detection method is applied from the user side, it is not currently applicable to mobile in-app ads [42, 45].

Gabryel presented a fraud detection algorithm based on criteria collected from the advertiser's websites through a special Javascript component; these measured features try to capture the end user interaction with the website such as the number of pages visited, page scrolling, keystroke data, and other client side information [50]. Fallah et al. built a KNN-based click fraud classification technique

with an introduction of a counter field to overcome the high memory consumption associated with this classification method [53]. They evaluated the performance of their algorithm on a data set collected from a search engine called parsiijoo and focused on user session information such as user clicks in the search engine and his IP address [53]. Almeida et al. followed a set of verification rules, such as not having an IP from blacklisted IPs, having valid HTTP headers and other criteria, to flag publishers as fraudulent if they do not pass these set of rules [56].

While these studies proposed many solutions for click fraud detection, they did not offer a crowdsource-based approach that can benefit from the large-scale crowdsourcing view to accurately detect malicious publishers and victim advertisers attacked by their competitors. In addition, some of these methods are managed by ad networks without collaborating with advertisers, whereas others are managed by advertisers without collaborating with ad networks.

Unlike previous works that offer a solution either managed by advertisers or ad networks, our work features a new fourth party called CFC (Click Fraud Crowdsourcing), trusted by both ad networks and advertisers, which main objective is to detect malicious clicks by crowdsourcing multiple ad click requests from different advertisers. Table 1 summarizes the key features in the click fraud literature and compares them to our work (CFC). Although our approach does not investigate existing click fraud attacks and threats, it offers a solution that protects both advertisers and ad networks.

2.2 CPA charging model literature

In recent studies, two approaches have been followed in terms of evaluating the CPA charging model, its advantages, concerns, and possible enhancements.

2.2.1 CPA—general overview

The first approach presents an overview of the CPA charging model, its advantages and disadvantages, and compares it with the CPC and CPM model. Studies in this category did not address security concerns in the CPA model, but rather economic concerns.

Pechuán et al. [22] presented an overview of the CPA model, its advantages and disadvantages. Mahdian et al. [23] also explained how the CPA model works, its framework, its advantages, and the challenges that it faces (in terms of feasibility, user privacy, and so on). Rosales et al. [24] provided an analysis of conversion rates in CPA or CPC models (where conversion is not guaranteed as opposed to CPM). By analyzing ad traffic logs from an ad exchange company called YAHOO's Right Media Exchange (RMX), they proved how the ad size directly affects the click-through-rate CTR (rate of ads being clicked after display) but not the conversion-rate CVR (rate of users performing actions in CPA), whereas the parameters age and gender affect CVR but do not affect the CTR.

Many studies addressed the CPA charging model from an economical approach: Hu et al. [25] evaluated from an economical point of view, the benefits and costs of the CPC and CPA model to both advertisers and publishers, by applying an economic framework that measures many key elements such as the ratios of purchases to clicks. Ross et al. [26] proposed a new approach that follows a combined contract of CPA (including CPC) and CPM models that can be financially beneficial to both publishers and advertisers. Dellarocas et al. [27] also explained several economic concerns of the CPA model. Hu et al. [28] suggested that the optimal contract between the advertiser and publisher that encourages them both to enhance their advertising efforts should be a combination of the CPM, CPC, and CPA model.

2.2.2 CPA—security overview

The second approach addresses different security concerns of the CPA model, and some studies propose solutions to these threats. Pechuán et al. [22] presented briefly many types of CPA scams such as cookie stuffing where in the context of web advertising, the publisher leaves cookies in the user's browser without the consent of the latter, in order to falsify a user's visit to the advertiser's website. They also

Table 1 Literature review summary vs. our work

Ref.	Offer click fraud tool trusted by ad networks	Offer click fraud tool trusted by advertisers	Investigate existing click fraud attacks and threats	Offer click fraud tool managed by a party trusted by both ad networks and advertisers
[6, 7–11, 12, 18, 34, 36, 38, 40, 43, 46, 49, 51, 52, 55, 57, 58, 59]	No	No	Yes	No
[3]	No	Yes	Yes	No
[13, 14–16, 17, 35, 37, 41, 44, 54]	Yes	No	No	No
[18, 19, 42, 45, 50, 53, 56]	No	Yes	No	No
CFC	Yes	Yes	No	Yes

proposed possible solutions/improvements for CPA scam detection; however, they did not present any detailed explanation to support their proposed solutions. A major concern in the CPA security field is the possible misreports of actions by advertisers: Agarwal et al. [29] highlighted several risks of using CPA pricing model, including the misreports of actions by advertisers to reduce advertising campaign costs.

Studies in this domain primarily focused on the bidding auction system employed in this charging model to determine the ad to be shown for a given ad slot. Mahdian et al. [23] explained a rank-by-revenue bidding model used to determine which advertiser's ad will be displayed in an ad slot. The advertiser's chance of winning an ad bid increases whenever he reports an action, although he is charged for each reported action, it might reduce his incentive of misreporting actions. Nazerzadeh et al. [21] proposed a mathematical bidding representation that can be used in the CPA pricing model. Similarly to [23], they demonstrated how their approach limits the incentive of dishonest advertisers when reporting actions.

Ding et al. [20] proposed a simple solution for detecting advertisers who under-report actions on their websites (to reduce their campaign costs in the CPA model). Their approach is based on a comparison between the advertisers' reported actions, and feedback from volunteered users who performed actions on these websites after clicking on their ads.

While the rank-by-revenue bidding model potentially reduces the advertisers' incentive of under-reporting, it does not offer a guaranteed metric to determine how much an advertiser should be charged (and thus how much a publisher should be paid), since the advertiser might preserve a balance between his campaign costs and his ranking, by reporting some actions and neglecting to report others.

Therefore, a secure measuring system that does not rely on the advertiser's input is required. Furthermore, as explained earlier, the actions used in the current CPA model might not reflect true users' interests in many cases, thus we propose a new mobile ad charging model that benefits from our CFC system to charge advertisers based on the duration spent in the advertiser's website. Table 2 compares our proposed CPA model with the literature.

3 Proposed solution

Our proposed system adopts a client-server architecture. It is composed of four components as shown in Fig. 2.

The first component represents the client side, which consists of ad banners that are shown in mobile applications. Since they are accessed via mobile phones, we represented them as mobile devices in Fig. 2. We will refer to this component as "ad banner component."

The server side contains three components: the first server side component consists of ad network APIs that handle the ad selection and billing for both publishers and advertisers. We will refer to this component as the "ad network component." The second server side component consists of advertisers' websites. We will refer to this component as "advertiser component." The third server side component consists of our online server that works as a click fraud detection engine. We will refer to this component as "CFC component—server side."

3.1 Ad banner component

This client side component is added to the publishers' applications by publishers that wish to generate revenues from ad clicks. These applications represent any application that can be downloaded and used by end users (usually via Play Stores).

In order to add the ad banner component to a mobile application, the publisher should sign up on an ad network website that he chooses and follow the instructions explained by the ad network to integrate this component in his application. This integration might vary from one ad network to another, but in general, it consists of downloading a jar file and adding it to the application with the publisher's credentials for this ad network service. This integration process is common for all ad networks; however, the only difference in our proposed model is that the downloaded jar file will actually contain, not only the ad network ad managing logic, but also our own CFC click fraud detection logic (which is explained in detail in Section 3.4). Furthermore, this addition to the CFC model is transparent to the publisher since no additional steps are required from him compared to ad banner integration with traditional ad networks that do not offer the CFC component as part

Table 2 CPA literature vs our work

Ref.	Present CPA overview (no security)	Present CPA overview (security)	Present solution controlled by advertiser	Present solution controlled by party trusted by advertiser and publisher
[25]	Yes	No	No	No
[23, 24]	Yes	Yes	Yes	No
[30]	No	Yes	No	No
[21, 22]	No	Yes	Yes	No
CFC	Yes	No	No	Yes

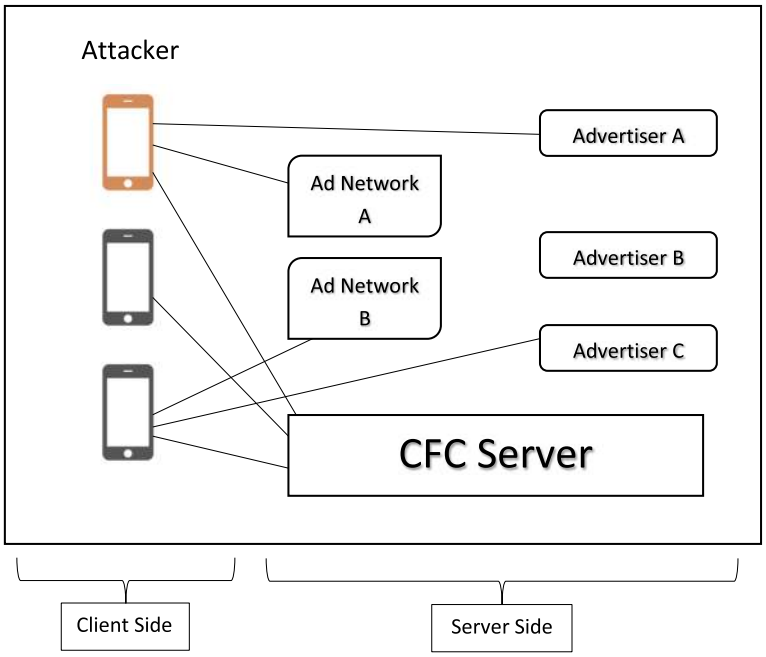

Fig. 2 CFC proposed solution overview

of their ad banner logic. It does not impose as well any extra effort on the end user side when viewing or clicking the ad banner in the mobile application.

Once added to the application, the ad banner will use APIs provided by the ad networks, to fetch ads and display them in the banner. The ads are then visible to end users who are using the publisher's application. Once the end user clicks on a banner ad, he will be redirected to the advertiser's website. At the same time, the ad banner component will be sending click information to the CFC server side component. These click information are explained in detail in Section 3.4.

3.2 Ad network component

Similarly to traditional ad networks currently in the market, this component acts as a relay between different publishers and advertisers, by selecting which ads to send to the publisher for display, charging advertisers for each ad click and paying the publisher a percentage of the charged money.

In order to give more assurance to the advertisers that the clicks to their websites are actually legitimate clicks, it is in the interest of ad networks to inform advertisers that they are using the CFC detection mechanism. To achieve this CFC protection, ad networks can request from the CFC party this service. This request can be made by email or using a website built by the CFC for this purpose. The CFC party will then send an encrypted jar file to the ad network, which can then be added by the ad network to the existing ad banner jar file that they offer to the publisher. Both files will be sent to the publisher. In other words, the ad network will merge the logic that they normally use to display ads in ad banners with the CFC logic that detects malicious activities. Although the CFC logic is not built and managed by the ad network, it will host it in its own ad banner. The implementation of the jar file that contains the ad banner logic is explained further in detail in Section 4 below.

3.3 Advertiser component

After a user clicks on an ad featured by the ad banner component, she will be redirected to the advertiser's website, which represents the advertiser component. No modification will be made to this component in our proposed model in comparison to traditional advertiser's websites.

3.4 CFC component (server side)

Similar to companies that offer anti-virus malware detection services, this component is managed by another party that does not benefit from clicks but rather present for the advertiser as a safety control mechanism. After collecting a large enough number of clicks from different ad banner components using the CFC service, a crowdsource-based calculation is performed. We will refer to this click fraud crowdsourcing algorithm as "CFCA." This calculation is following a *crowdsource* approach by building its analysis results on a large number of clicks in order to reduce false positives for a given user. For example, if the user in question is practicing suspicious behavior on several applications or on several

occasions, it is highly possible that he is in fact malicious. The more we have records of his suspicious activities, the more our judgement can be certain, and this is where we benefit from our crowdsource approach.

In addition to following a crowdsource approach, the motivation behind our approach is based on two main needs: on the one hand, an ad network is able to monitor and assess many clicks from different apps; however, it is not able to monitor the user's activity for click fraud detection once she is redirected to the advertising site; on the other hand, while an advertiser is able to monitor the user activity on her site (for example the duration spent on the site), the judgment is done per click and is therefore prone to a high false positive rate [3]. Accordingly, if a user clicks on an ad and she shows no interest in the ad website, she will be flagged as malicious for quickly exiting the website. Our proposed solution addresses these two shortcomings by combining both the click information provided by the ad networks and the user activity information provided by the advertiser. In addition to the CFCA, the CFC party manages several APIs that communicate with the CFC ad banner component and a corresponding online database.

3.5 CPA enhanced model

Our proposed mobile ad charging model benefits from our CFC system to charge advertisers based on the duration spent in the advertiser's website. As opposed to current CPA systems that rely on the advertisers to report the actions to be charged for, our CFC model presents a secure framework in which the duration is measured by a trusted party. In addition, our system reduces the work load on the advertisers since they are not required to perform any integration on their websites or perform any action reporting to the ad network.

For billing purposes, the CFC must send the captured actions (ad request with duration information) to the corresponding ad network. The ad network can define the pricing scheme for different durations, for example, charge the advertisers if the duration spent on the advertiser's website by the user is more than 1 min or even charge as a function of the time spent.

Although this new mobile ad charging model is represented as an enhancement to the CPA model, it can also support the CPC and CPM models: The CFC system registers whenever an ad is clicked, which can be used as a CPC model. In addition, it can keep track whenever a new ad is shown in the application, which can be used as a CPM model. In fact, the CFC system could be used as a combination of the three models (CPA, CPC, and CPM).

3.6 Ad banner component CFC steps

Although the application can be used by the publisher, the CFC steps are applied to any user that uses the application and interacts with the ad banner component (as explained in this section); we will refer to this user with the mobile phone symbol in Fig. 3. When the application starts, our library requests an ad to show from the ad network by sending the publisher's ID (Fig. 3—step 1). Unlike traditional ad fetching systems where each ad network manages its own publishers' identification system, this ID is generated by the CFC party and given to the publisher by the ad network. This is to ensure that each publisher has a unique identifier among all the ad networks registered with the CFC services.

As a response to the ad request, the ad network returns information about a selected ad. The ad banner component then displays the ad in a simple banner.

When the user clicks on an ad, the CFC ad banner component performs several sequential requests (Fig. 3):

In steps 3–4, for billing purposes, the library informs the ad network that an ad is clicked, by sending the publisher's ID and the ad ID (Fig. 3—step 3). After confirming this publisher-ad ID combination, the ad network returns a confirmation billing response to the publisher (Fig. 3—step 4).

Steps 1, 2, 3, and 4 are performed similarly to existing ad fetching systems. However, the following steps are added in our system:

In steps 5, 6, and 7 (ad clicked challenge), we consider, intuitively, an ad click as potentially fraudulent, if after clicking on the ad and being redirected to the advertiser's landing webpage, the user spends less than a certain time before exiting the advertised website. Therefore, the CFC ad banner component sends a request to the CFC server to indicate that an ad view session has begun on the client side. This request takes as input in step 5, the publisher's ID and the user's IP address, a timestamp (of when the ad was clicked), and a state integer of value 1. This non-local IP address is fetched on the client side using an online service called "ipify" [32]. The CFC server saves this information in its online database with a state field of value 1 and a timestamp. The server saves the timestamp for future reference.

In step 6, and to verify that the extracted IP is not spoofed, the CFC server challenges the client side by sending a random token and a created session ID. To prove its IP address legitimacy (step 7), the client sends back the challenge token with a state equal to 2, and the session ID. The state field identified by the returned session ID is updated to 2 in the online CFC database (after verifying that the previous state value of this session is 1). The state field is used to keep track of the actions performed by the client side; for example, a state of value 2 means that the user has clicked on the ad but has not exited the advertised website yet.

In step 8 (ad view), after receiving a confirmation of the ad click challenge from the server, the user is redirected to the advertised website.

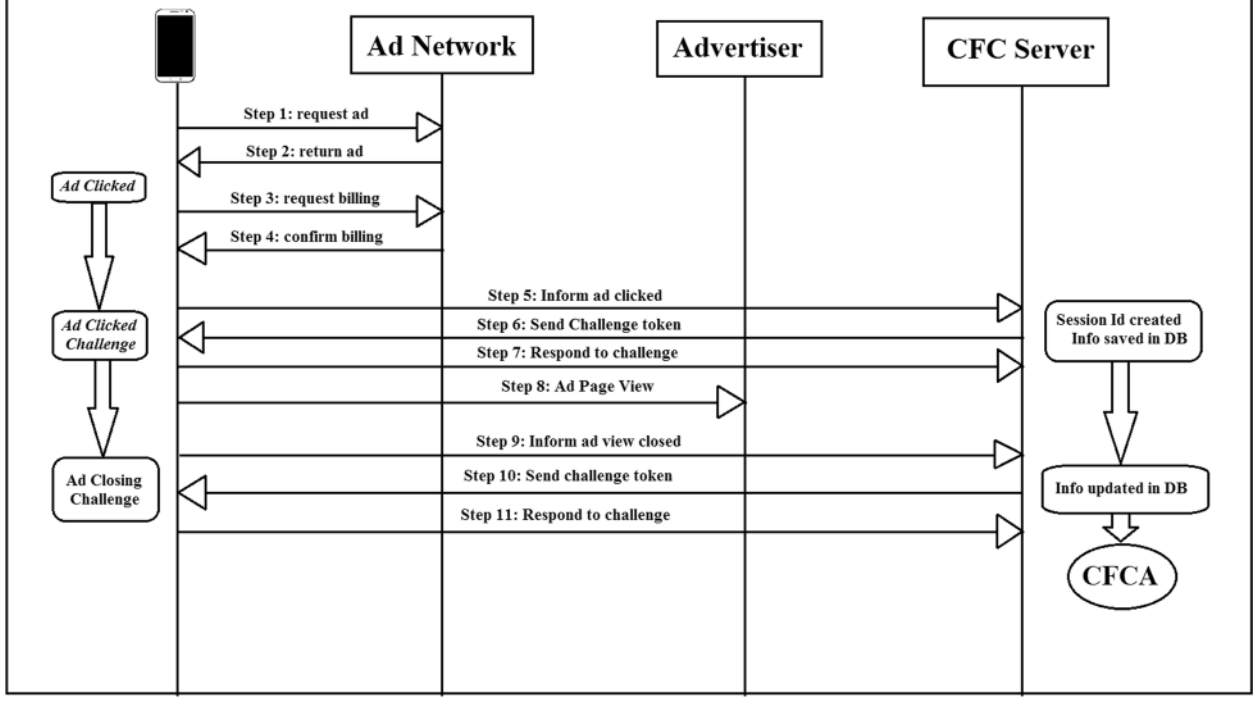

Fig. 3 CFC steps

In steps 9, 10, and 11 (ad closed challenge), once the webview is closed, or the application is no longer visible (by detecting when it is no longer in foreground after clicking the exit button), the client library informs the CFC server of the ad view session ending, by sending the session's ID, publisher's ID, user's IP address, the new timestamp (of when the ad was closed), and state of value 3 (step 9). The CFC server then updates state value of the record identified by the session's ID to 3 (after verifying that the previous state value of this session is 2).

Similarly to the ad clicked challenge explained in steps 6 and 7, to check if the client's IP is spoofed, the server generates a new random challenge token and sends it back to the client in step 10. To prove that its IP is not spoofed, the client sends back the new challenge token with the session's ID, publisher's ID, IP address, and a state of value 4.

After verifying that the credentials sent by the client are correct and that the previous session state value is 3, the CFC server decides whether to consider this ad session as potentially malicious or not based on one/multiple criteria set by the CFC admin such as the difference between the previous saved timestamp and the current received timestamp, which represents the duration spent on the advertiser's website. The ad requests that are flagged as potentially malicious, are updated in the database with a state value of 5. The ad requests that are flagged as non-malicious are updated in the online database with a state of value 4. Ad requests saved in the online database of both states 4 and 5 are added to the CFCA for analysis.

3.7 CFCA component (server side)

Besides managing the APIs and the online database, the server performs the CFCA in order to identify malicious publishers. The intuition behind our algorithm is based on the following idea: It is likely that a legitimate app (associated with a publisher) will have many non-malicious users that clicked on an ad and exited the ad webpage simply because they were not interested anymore in the landing page. However, it is unlikely that a non-malicious app has a high number of these suspicious requests. To reduce the false positive rate, we compare the percentage of malicious clicks per publisher to a starting point determined by the CFCA admin.

This system benefits from both a global view, where it gathers multiple ad requests data corresponding to different ad network-publisher-advertiser combinations, and a local view, where it is able to track the user's engagement in each advertising website.

3.8 Attacker's model

To evaluate the robustness of our click fraud detection system, we tested many different attackers' models whose goal is to generate high revenues from ad networks:

- Type A: A malicious publisher (without IP spoofing) creates a repetitive click automated tool that does not spend a long time on the advertiser's website, since it is forced to exit the ad to open another ad immediately. This attack is easily detected by the proposed system since the duration spent on the advertiser's website is calculated and it is used to determine whether the publisher is malicious or not.
- Type B: A malicious publisher (without IP spoofing) places ads next to buttons in order to trick user into clicking them. This fraudulent act is known as placement ads in the literature. Since such a user is not necessarily interested in the ad, she will most likely exit the ad webpage directly, and thus this small session duration will be flagged by the proposed system as potentially malicious. However, there is a slight chance that, although tricked into clicking it, the user spends more than minimum required time on the advertised website. We consider these redirected ad requests as non-malicious since the users could become customers from the advertiser's point of view.
- Type C: A malicious publisher (without IP spoofing), after completing the ad clicked challenge, closes the webview in less than the minimum number of seconds, to be able to open a new ad session. However, being on the client's side, she is able to drop the request generated by the CFC ad banner component when the ad is closed (when the duration spent is less than the minimum number of seconds), and fabricate this request later, after the minimum number of seconds has passed. The goal of this attack is to be able to repeatedly click on ads to generate higher revenues without spending the required duration on the advertiser's website and without being detected. Although the attacker is able to drop the legitimate ad closing request and fabricate it, and by that falsify the duration spent on the advertiser's website, however, since the IP is not spoofed, the CFC server can identify the user and therefore limit this attack to just one undetected ad request. If the malicious publisher fabricates many falsified ad requests, the CFC can detect abnormal entries of the same IP. For example, it is not feasible for the same user identified by IP, to spend more than a minimum number of seconds on 3 different websites in a short time interval.
- Type D: A malicious publisher uses a spoofed IP address to be considered as a new user with each ad request. Whether this publisher fakes IP before or after completing the first ad clicked challenge (steps 5, 6, 7), since she did not complete both challenges, her state in the online database will not be updated to the final state of the ad requests considered as non-malicious. As explained in Section 3.7, the CFCA can simply time-out the ad requests with a state different than the final state and a reasonably old timestamp (to take into consideration honest sessions that still have not completed both challenges).
- Type E: A malicious publisher hires multiple human clickers to imitate the normal user's behavior by clicking on the ad and spending enough time to avoid being detected by our system. In addition to being forced to spend a significant number of seconds on each advertising website, the proposed system is able to identify the human clicker by her IP address that is sent with each ad click. When a high number of click requests with the same IP address is detected, the CFC server will flag the publishers using this IP address as potential threats. This limits the number of fraudulent clicks per human clicker before being detected as malicious by our system.

4 Implementation

4.1 Ad banner component CFC steps

Using Android Studio, we built the CFC ad banner component as an exported jar file. We used an obfuscation tool, "Proguard" [30] that obfuscates the jar file targeting for example class and variable names. We inspected the effect of this obfuscation using the decompiler tool, "Java Decompiler" [31]. As expected, we were not able to reconstruct the classes in our jar file by reverse engineering. To test its functionalities, we created a sample publisher's Android application (Fig. 4) that hosts ads using this library, in order to generate a revenue on every ad click.

Using the LAMP server [23], We built the API request_ad(publisher_id), managed by the ad network, that takes as input the publisher's ID and returns information about a selected ad in JSON format (Fig. 3—step 2). After fetching the ad, the library parses the JSON response and displays the ad on the publisher's application in a simple banner (Fig. 3—step 2).

In current ad fetching systems, ad networks follow an ad selection process where they decide which ad to send for display based on factors such as the bidding placed on the ad by the advertiser, application specific targeting mechanism, whether the ad was already displayed in the corresponding application, etc. However, this ad selection process is beyond the scope of this paper. For simplicity, we are generating a generic ad in the form of a textual title, a textual descriptive content, an ad ID, and a corresponding ad URL.

To redirect the user to the advertised website after an ad is clicked, we used the Android native component

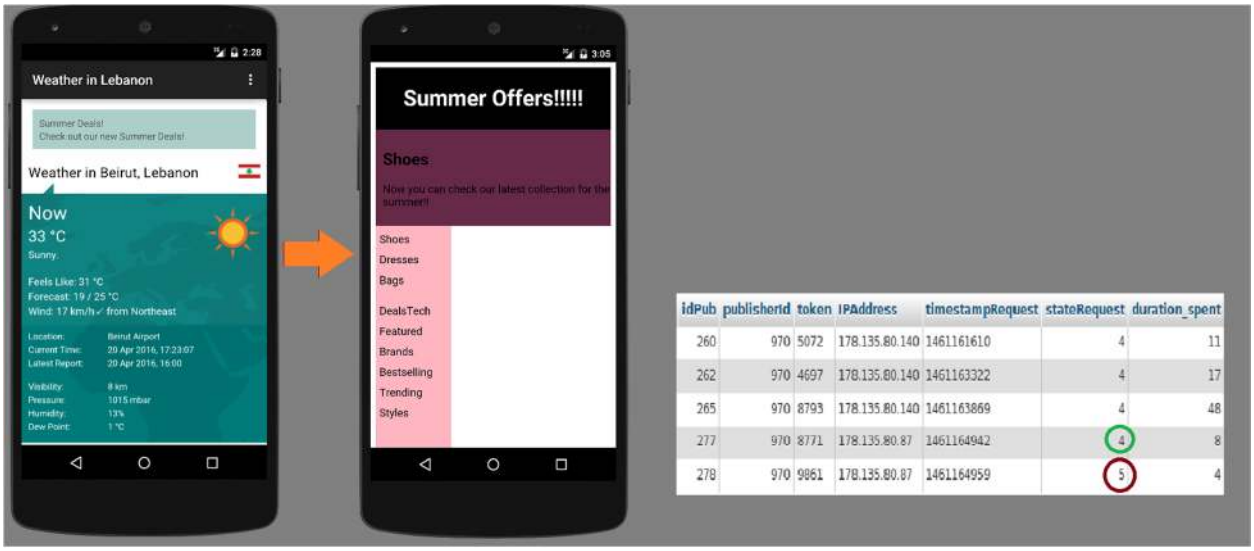

Fig. 4 CFC client and online database

known as a webview. This webview is called and managed by the phone CFC component. The benefit of using a webview in this proposed system is the ability to detect when the user exits the advertiser's website, by either clicking the native Android back button, or by clicking the exit button. Webview requests the corresponding ad URL and loads the advertised website. We created a sample advertiser's website to test this step.

4.2 CFCA dataset

To evaluate the performance of our algorithm, we generated the following dataset using MATLAB: a population of ad clicks of size N in a given period. An ad click in this context represents a session in which the user clicked on an in-app ad, viewed the advertiser's website for a certain duration, and closed the ad webview after performing the handshake, i.e., ad request that completed the 11 CFC steps with state = 4 (duration > 5 s) or state = 5 (duration < 5 s). By detecting publishers who used spoofed IPs based on the saved state in the online database (state = 1, 2, or 3), we are able to immediately filter these suspicious publishers. Therefore, we do not need to take them into consideration in our CFCA duration-based algorithm.

The created N ad requests correspond to a total number of publishers, TNP. We set a fixed percentage of malicious publishers, MPP. We consider a publisher to be malicious if she performs any type of click fraud by repeatedly clicking on ads and exiting the advertiser's website to generate further revenue. We consider an ad request to be suspicious if the state saved in the online database corresponding to this instance is equal to 5, which means the duration spent on the advertiser's website is less than or equal to 5 s. For each of these malicious publishers, we set a random percentage of suspicious clicks, RMC, such that RMC is larger or equal to a truth starting point:

$$\mathrm{TrustST}\,(\mathrm{RMC} >= \mathrm{TruthST})$$

This TruthST is the lower limit used to classify publishers as honest or malicious in the created dataset. In other words, if 30% of a publisher's clicks are suspicious, then this publisher is malicious (30% > TruthST). This classification is considered as the ground truth definition which can be configured to be different. The total number of ad requests per malicious publisher is TNMP.

For each of the honest publishers, we set a random percentage of suspicious clicks, RHC, such that RHC is less than the truth lower limit TruthST (RHC < TruthST). The RHC number shows the number of clicks clicked by a valid user who had the intention of viewing the advertiser's website but did not spend more than the minimum number of seconds in the website. The total number of ad requests per honest publisher is TNHP.

4.2.1 CFCA classification

To simulate ad traffic in a real setting, we performed multiple iterations; each iteration corresponds to a time slot in which a random number NR of samples (ad requests) is taken from the total population N without replacement. In a real scenario, each iteration corresponds to an instance when the CFC administrator fetches ad requests from the online database. These samples are used as input to CFCA in order to classify the publishers as honest/malicious.

In each iteration, we classify each publisher based on a CFCA starting point CFCAST as follows: (1) honest, if the percentage of suspicious clicks from the total ad

requests of this publisher does not exceed the CFCAST, (2) malicious, if the percentage of suspicious clicks from the total ad requests of this publisher exceeds the CFCAST, and (3) not classified, if the total number of ad requests of this publisher is less than MinNbre (to have statistically significant data).

4.3 CFCA dataset evaluation

To evaluate the efficiency of our method under different scenarios, we created the following base set: the total number of ad requests N = 500,000, the total number of publishers TNP = 500, and the truth lower limit used for classification TruthST = 25 (different than CFCAST).

We used a total number of requests per malicious publishers 5000 < TNMP < 6000, and a total number of requests per honest publishers 1 < TNHP < 1500, with a minimum number of request to be able to classify a publisher as MinNbre = 100. Note that TNMP is higher than TNHP because clicking on an ad is considered as a rare event in the mobile advertising industry [33], which makes the number of expected clicks to be low. Therefore, if the total number of clicks is high, then it is most likely that a high percentage of it is malicious.

We evaluated our model under different scenarios: we tested three different percentages of malicious publishers MPP = 5, MPP = 10, and MPP = 15. For each of these scenarios, we tested our CFCA using different CFCA starting points, CFCAST = [10, 20, 30, 40, 50, 60, 70, 80]. We calculated the false positive rate (FPR) (Fig. 5), true positive rate (TPR) (Fig. 6), and accuracy (ACC) (Fig. 7) in each iteration.

As part of their proposed methodology, the authors of [3] used the duration spent on the advertiser's website to detect malicious clicks on a per click basis, which means that every ad request that results in an advertisement view of less than 5 s for example is considered as malicious. This duration is calculated by the advertiser and thus cannot be trusted by the publisher. In addition, as mentioned before, it is likely that a legitimate app generates an honest ad click request without spending more than 5 s on the advertising website because the user lost interest in the landing page. However, it is unlikely that a non-malicious app has a high number of these requests.

We compared our proposed method with the approach proposed in [3] by using our CFCA algorithm with our base set, different CFCA starting points, CFCAST = [10, 20, 30, 40, 50, 60, 70, 80], and a starting point of 1 for their results (to simulate their per-click approach). We calculated for the different starting points, the FPR (Fig. 5), TPR (Fig. 6), and ACC (Fig. 7) in each iteration.

5 Results

The results presented in Figs. 5, 6 and 7 correspond to the iteration number I = 25.

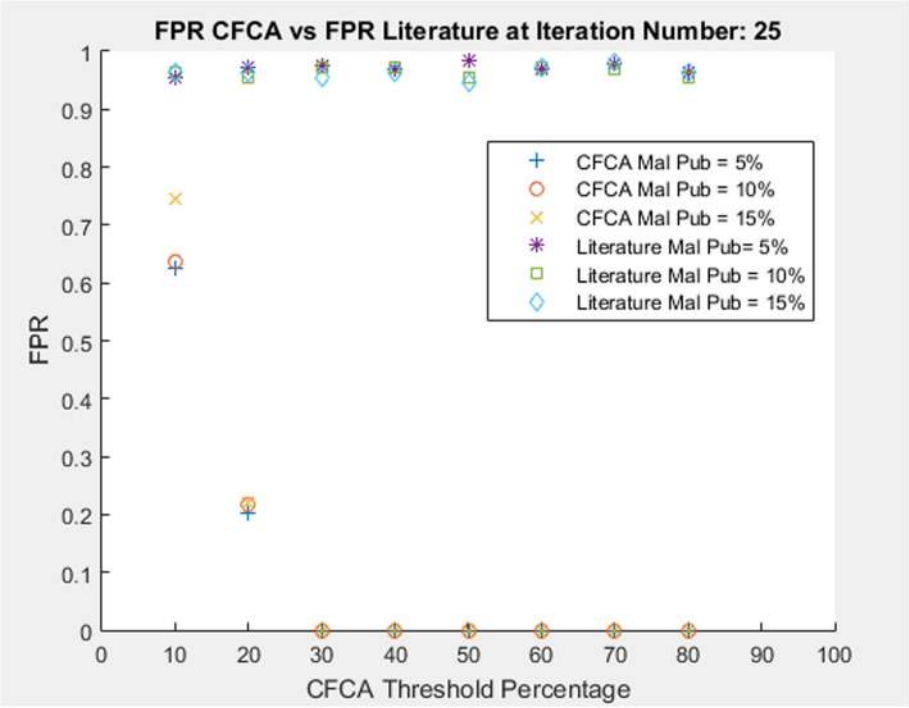

Fig. 5 FPR CFCA vs FPR literature at iteration I = 25

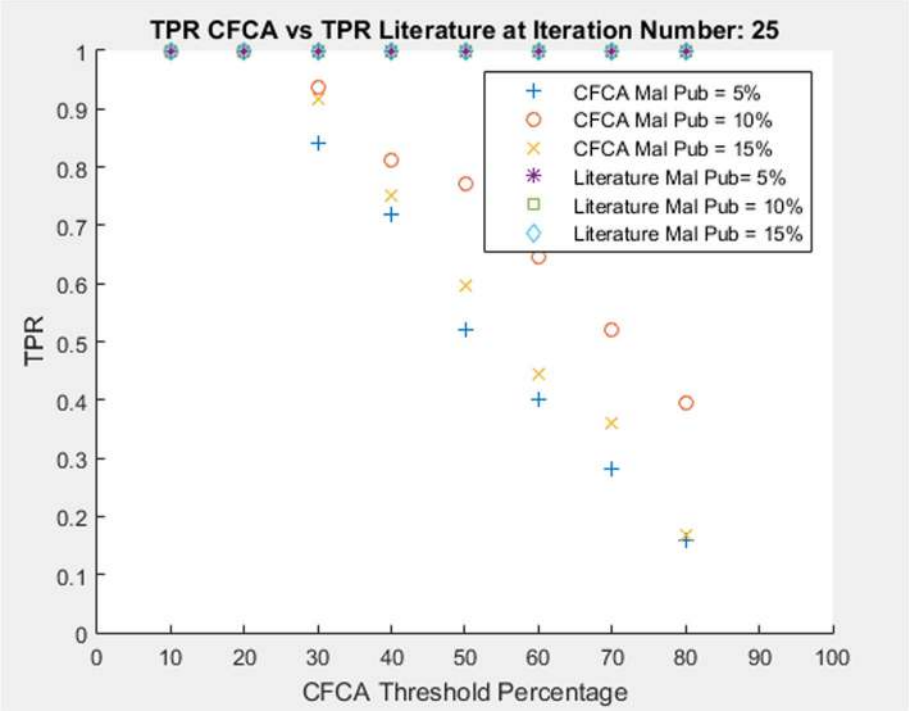

Fig. 6 TPR CFCA vs TPR literature at iteration $I = 25$

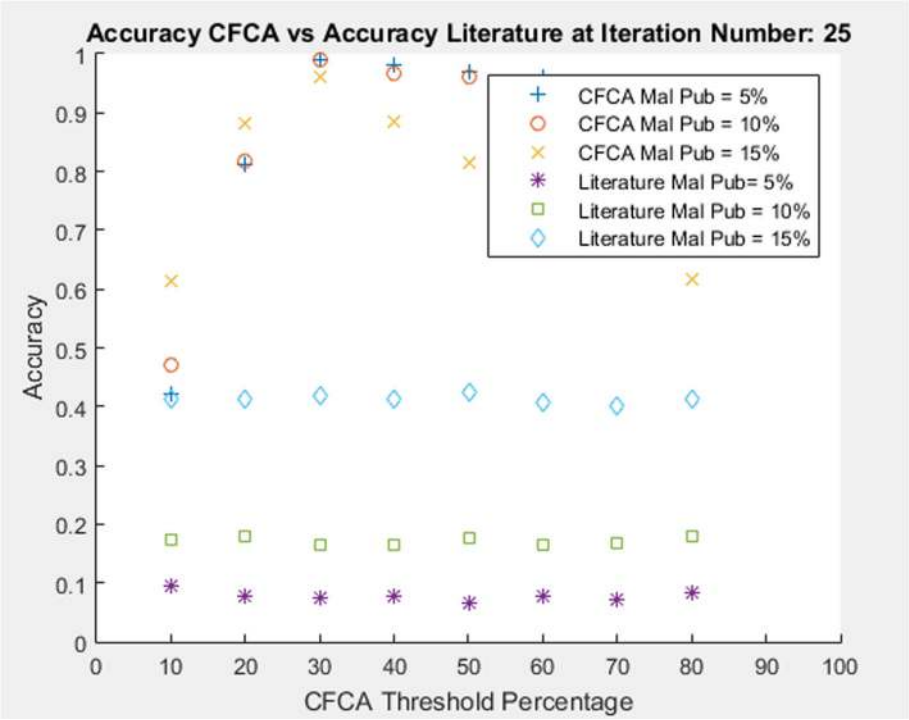

Fig. 7 Accuracy CFCA vs accuracy literature at iteration $I = 25$

Based on the results in Fig. 6, for a percentage of malicious publishers (MPP) = 10, our model presents a low false positive rate starting from FPR = 0.21 when using a CFCA starting point of CFCAST = 20, and it decreases to FPR = 0 with the increase of the CFCA starting point to CFCAST = [30, 40, 50, 60, 70, 80]. As expected, the FPR calculated based on the starting point of 1 used in [3] by adopting a starting point = 1 presents a very high false positive rate with an average of FPR = 0.964 for the different CFCAST scenarios.

Based on the results in Fig. 7, our model presents a high true positive rate, TPR = 1, when using a CFCA starting point of CFCAST = 10 or CFCAST = 20. The TPR is reduced to TPR = 0.937 with the increase of the starting point to CFCAST = 30. As expected, it decreases with the increase of the CFCAST continuously, whereas the TPR calculated based on the starting point of 1 used in [3] remains TPR = 1 for all the different scenarios.

Based on the results in Fig. 8, our model presents a high accuracy, ACC = 0.82, when using a CFCA starting point of CFCAST = 20, and it increases to ACC = 0.98 with the increase of the CFCA starting point to CFCAST = 30. It maintains this high level of accuracy for the rest of the CFCA starting points.

6 Analysis

Based on the results, we can conclude that changing the MPP did not highly affect the results, except when measuring the accuracy in each iteration, whereas the effect of changing the CFCAST is visible in the figures.

Therefore, in order to determine the appropriate CFCA starting point to use, we generated the ROC curve for iteration $I = 25$ using our base set with a malicious publisher percentage of MPP = 10 (Fig. 7). Depending on how aggressive we would like our model to be, we can select between two CFCA starting points, CFCAST = 20 (TPR = 1, FPR = 0.21) or CFCAST = 30 (TPR = 0.93, FPR = 0). Another method to determine which starting point to use is the F-measure (also known as F1 score) that takes into consideration the precision (also known as positive predictive value) and the recall (also known as the sensitivity or TPR). The F1 score when using CFCAST = 20 is F1Score = 0.6532 whereas it is F1Score = 0.967 when using CFCAST = 30.

To evaluate how fast our proposed model is able to achieve a high true positive rate while maintaining a low false positive rate and a high accuracy level, we tested it at different iterations ($I = 5$, $I = 6$, $I = 7$, $I = 8$, $I = 10$ …). We concluded that our system converges starting at iteration $I = 7$ (Figs. 9, 10 and 11), whereas it presents Null FPR values at earlier iterations ($I < 7$). It is expected not

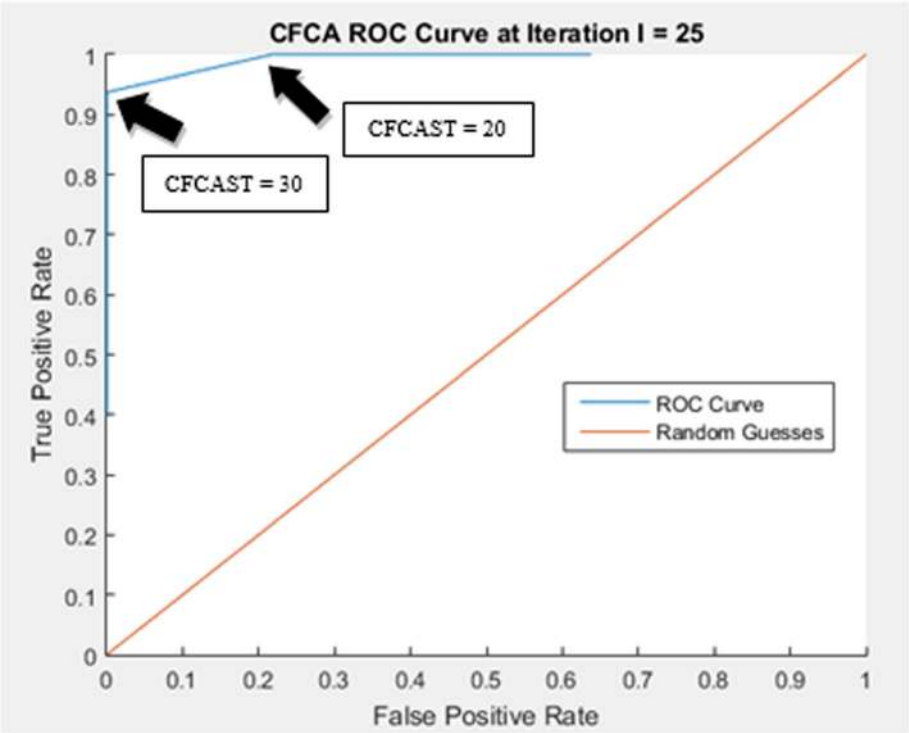

Fig. 8 CFCA ROC curve

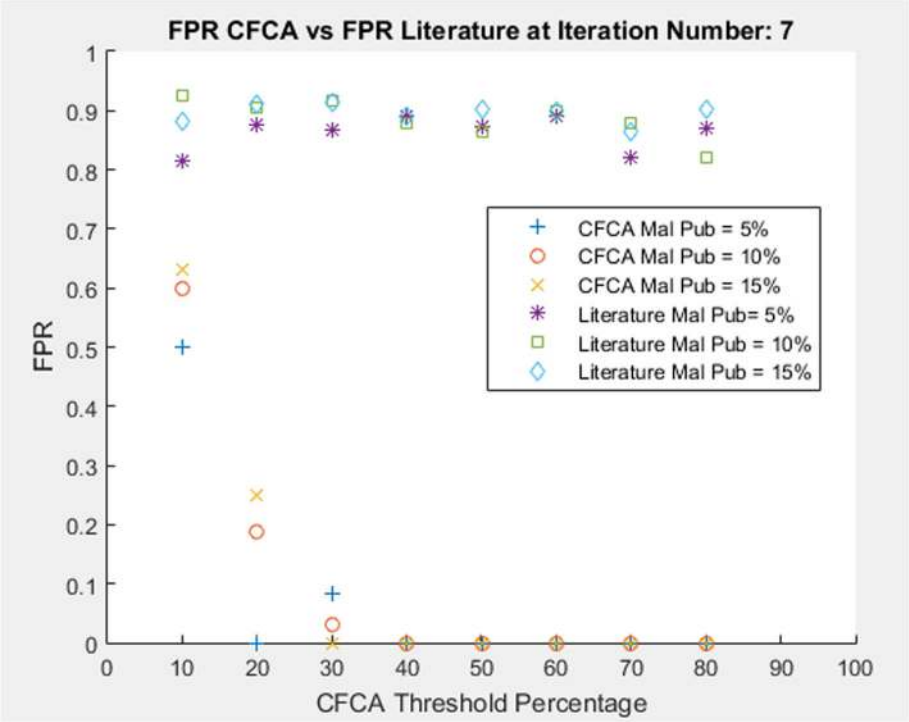

Fig. 9 FPR CFCA vs FPR literature at iteration 7

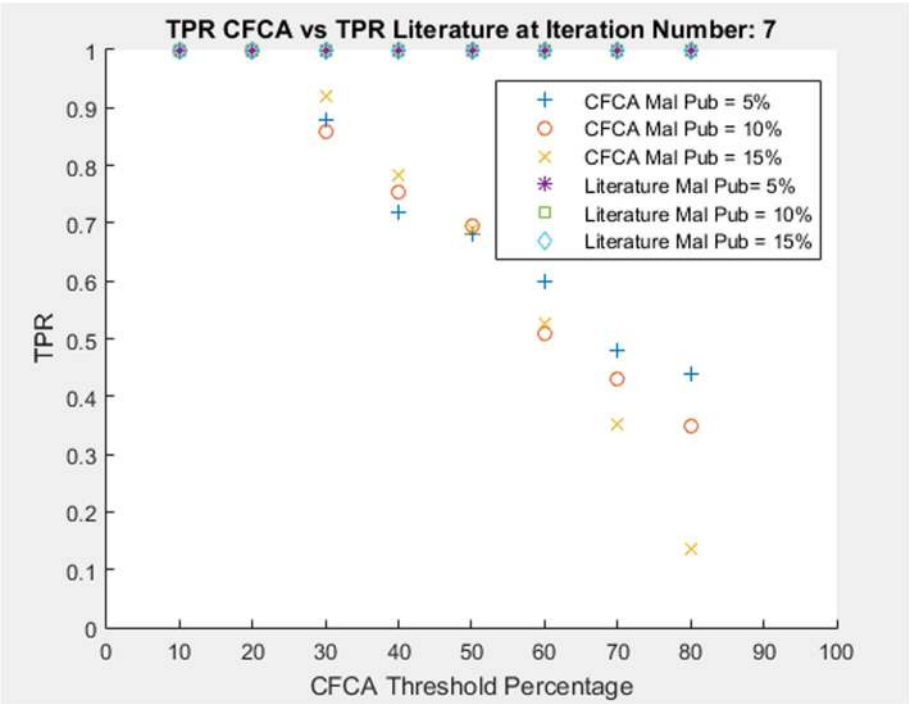

Fig. 10 TPR CFCA vs TPR literature at iteration 7

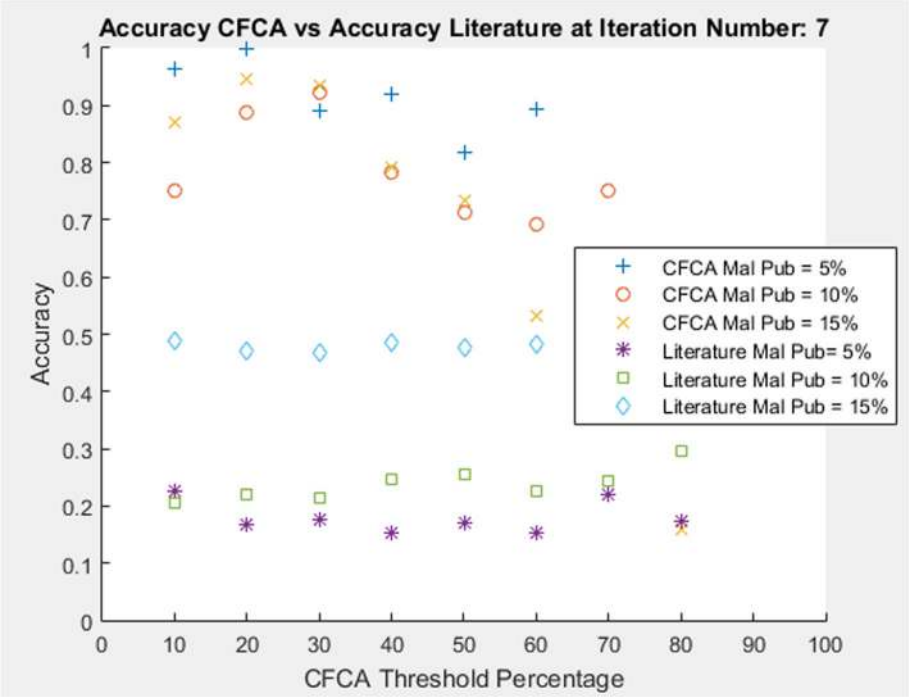

Fig. 11 ACC CFCA vs ACC literature at iteration 7

to have any FPR values at very low iterations ($I < 7$) because FP = TP = 0 in this case since we set a minimum of 100 requests per publisher to be able to classify him.

To measure the accuracy of the duration captured by the CFC ad banner component, we clicked on the ad webview in our Android application (Fig. 4), closed the ad after registering manually the duration of this ad view (not using CFC), and compared it with the duration saved in the online database (calculated by the CFC ad banner component). We repeated this experiment 30 times for a duration less than 5 s, and 30 times for a duration higher than 5 s and calculated the accuracy of the duration in each of these experiments. As shown in Table 3, the accuracy of the duration measured by CFC in both types of the experiments is very high (85.46% and 90.73%).

To measure the resource consumption of our CFC ad banner component, we created a sample Android application that displays a white background in our CFC banner ad. At run time, the ad banner component fetches an ad and displays it in the banner ad. We clicked on the ad, spent 5 s on the advertiser's website, and closed the ad. The goal of this experiment is to measure the bandwidth and battery consumption caused by the CFC ad open/close challenges or any other component implemented in our system that are not adopted by traditional ad networks such as the opening/closing of webview. Therefore, we used an experiment duration of 30 s that is sufficient to be able to achieve these two handshakes, ad webview opening and ad webview closing, regardless of the duration spent on the advertiser's website. We do not address the resource consumed while browsing the advertiser's website since it depends on the content delivered by each advertiser's website. To have a more accurate result, we repeated the experiment 10 times (30 s duration each).

We created another Android application that integrates AdMob [33] banner ads. We repeated the same experimental procedure by clicking on the ad in each experiment. Unlike our CFC system, after clicking an ad, AdMob [33] either opens the Google Play Store in case the ad is an Android application ad or opens the default web browser in the phone such as Google Chrome.

Table 3 CFC duration accuracy

Duration	Average accuracy (%)	Average (s)	STDEV (s)
Less than 5 s (30 experiments)	85.461	0.497	0.0896
More than 5 s (30 experiments)	90.737	0.799	0.0653

Table 4 Resource consumption comparison

Ad network	Bandwidth (kb/30 s)		Battery (J/30 s)	
	Average	STD	Average	STD
CFC	18.494	5.098	0.7851	0.213
AdMob [14]	385	26.13	12.13	2.24
Google Play Store	144.44	154.68	3.03	1.348

Therefore, we evaluated also the bandwidth and battery consumption in Google Play Store or the default web browser (depending on the ad clicked) during these experiments. To have a more accurate result, we repeated the experiment 10 times (30 s duration each).

As shown in Table 4, our proposed model consumes, in 30 s, an average of 0.7851 J, whereas AdMob [33] consumes on average 12.13 J, and 3.03 J to display the advertiser's application in the Google Play Store.

Concerning bandwidth, Table 4 shows our proposed model consumes, in 30 s, an average of 18.494 kb, whereas AdMob [33] consumes on average 385 kb, and 144.44 kb to display the advertiser's application in the Google Play Store. Based on these experiments, we can conclude that both bandwidth and battery consumptions in CFC are very minimal compared to the popular ad network AdMob [33].

7 Conclusions and future work

We proposed a new crowdsource-based system that collaborates with both advertisers and ad networks in order to protect both parties from click fraudulent acts. This system manages ad fetching and ad clicks and monitors the activity of redirected users on the advertiser's website. It is able to track the user's duration in each advertising website and at the same time to gather multiple ad requests data corresponding to different ad network-publisher-advertiser combinations. In addition, the information gathered securely in our proposed model can be used as an enhancement to the CPA model.

Our results showed that our proposed method is able to lower the false positive rate (FPR = 0 with CFCAST = 30) when detecting click fraud as opposed to proposed solutions in the literature (FPR = 1) while having a high true positive rate (TPR = 0.93 with CFCAST = 30). Our system is transparent to both the publisher and the advertiser.

However, our framework suffers from three main limitations: it requires an additional step to be performed by the ad network to merge its library with the phone CFC library, it presents a privacy concern since the IP associated with the used application and timestamp is sent to the CFC server, and it requires the advertisers and ad networks to trust the CFC party.

In future work, we will implement a crowdsource-based algorithm that detects whether an advertiser is attacked by its competitors, and whether the user installed an advertised application in case she is redirected after clicking the ad, to an application download page instead of an advertising website.

Acknowledgements
Research was funded by Telus Corporation, Canada.

Authors' contributions
All authors read and approved the final manuscript.

Authors' information
Not applicable

Funding
Research was funded by Telus Corporation, Canada.

Availability of data and materials
Not applicable

Competing interests
The authors declare that they have no competing interests.

Received: 8 August 2018 Accepted: 2 July 2019
Published online: 22 July 2019

References

1. AppBrain, "Distribution of free vs. paid Android apps", [Online]. Available: http://www.appbrain.com/stats/free-and-paid-android-applications Accessed 29 Mar 2019.
2. Statistica, "Worldwide mobile app store revenues in 2015, 2016 and 2020 (in billion U.S. dollars) ", [Online]. Available: https://www.statista.com/statistics/220186/total-global-in-app-revenue-forecast/Accessed 29 Mar 2019.
3. V. Dave, S. Guha and Y. Zhang, 'Measuring and fingerprinting click-spam in ad networks', ACM SIGCOMM Computer Communication Review, vol.42, no. 4, p. 175, 2012, Helsinki, Finland.
4. dmnews, "Bots Mobilize", [Online]. Available: http://www.dmnews.com/mobile-marketing/bots-mobilize/article/291566/ Accessed 29 Mar 2019.
5. "The Alleged $7.5 Billion Fraud in Online Advertising", Online. Available: https://moz.com/blog/online-advertising-fraud Accessed 29 Mar 2019.
6. G. Cho, J. Cho, Y. Song, H. Kim, *An empirical study of click fraud in mobile advertising networks', in Availability, Reliability and Security (ARES), 2015 10th International Conference on (pp 382-388)* (IEEE, Toulouse, France, 2015)
7. J. Crussell, R. Stevens, H. Chen (2014). 'MAdFraud: Investigating ad fraud in android applications', in Proceedings of the 12th annual international conference on Mobile systems, applications, and services (pp 123-134), Bretton Woods, NH, USA, ACM.
8. T. Blizard,, N. Livic, (2012). 'Click-fraud monetizing malware: A survey and case study', in Malicious and Unwanted Software (MALWARE), 2012 7th International Conference on (pp 67-72), Puerto Rico, USA, IEEE.
9. SA Alrwais, A Gerber, CW Dunn, O Spatscheck, M Gupta, E Osterweil (2012). Dissecting ghost clicks: Ad fraud via misdirected human clicks', in Proceedings of the 28th Annual Computer Security Applications Conference (pp 21-30), Florida, USA, ACM.
10. B. Stone-Gross, R. Stevens, A. Zarras, R. Kemmerer, C. Kruegel, G. Vigna, *Understanding fraudulent activities in online ad exchanges,' in Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference* (ACM, Berlin, Germany, 2011), pp. 279–294
11. P. Pearce, V. Dave, C. Grier, K. Levchenko, S. Guha, D. McCoy, G.M. Voelker, *Characterizing large-scale click fraud in zeroaccess,' in Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security* (ACM, Arizona, USA, 2014), pp. 141–152
12. B. Miller, P. Pearce, C. Grier, C. Kreibich, V. Paxson, *What's clicking what? techniques and innovations of today's clickbots,' in Detection of Intrusions and Malware, and Vulnerability Assessment* (Berlin Heidelberg, Amsterdam, The Netherlands, Springer, 2011), pp. 164–183
13. B. Liu, S. Nath, R. Govindan, J. Liu, *DECAF: detecting and characterizing ad fraud in mobile apps* (Proc. of NSDI, Seattle, WA, USA, 2014)
14. A. Juels, S. Stamm, M. Jakobsson, *Combating Click Fraud via Premium Clicks* (USENIX Security, Santa Clara, CA, USA, 2007), pp. 1–10
15. D. Vasumati, M.S. Vani, R. Bhramaramba, O.Y. Babu, *Data Mining Approach to Filter Click-spam in Mobile Ad Networks,' in Int'l Conference on Computer Science, Data Mining & Mechanical Engg. (ICCDMME'2015) April 20-21, 2015 Bangkok (Thailand)* (2015)
16. W.Li, H. Li, H. Chen, Y. Xia (2015). 'AdAttester: Secure Online Mobile Advertisement Attestation Using TrustZone', in Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services. 75-88), Florence, Italy, ACM.
17. H. Haddadi (2010). 'Fighting online click-fraud using bluff ads',ACM SIGCOMM Computer Communication Review, 40(2), 21-25, New Delhi, India.
18. H. Xu, D. Liu, A. Koehl, H. Wang, A. Stavrou (2014). 'Click Fraud Detection on the Advertiser Side', in Computer Security-ESORICS 2014. 419-438), Wroclaw, Poland, Springer International Publishing.

19. M. Vani, R. Bhramaramba, D. Vasumati, O.Y. Babu, A Node-Tag Prediction Algorithm for Touch Spam Detection in Mobile Ad Networks. Int. J. Comput. Eng. Appl **VIII**(III), Part I (2014)
20. X. Ding, *A hybrid method to detect deflation fraud in cost-per-action online advertising* (Applied Cryptography and Network Security, Springer Berlin Heidelberg, 2010)
21. H. Nazerzadeh, A. Saberi, R. Vohra, in *Proceedings of the 17th international conference on World Wide Web*. Dynamic cost-per-action mechanisms and applications to online advertising (ACM, New York, 2008), pp. 179–188
22. L.M. Pechuán, E.M. Ballester, J.M.G. Carrasco, Online Advertising and the CPA Model: Challenges and Opportunities. Int. J. Eng. Manage. Res. **4**(3) (2014)
23. M. Mahdian, K. Tomak, Pay-per-action model for on-line advertising. Int. J. Electronic. Commerce. **13**(2), 113–128 (2008)
24. Rosales, Rómer, Haibin Cheng, and Eren Manavoglu. "Post-click conversion modeling and analysis for non-guaranteed delivery display advertising." Proceedings of the fifth ACM international conference on Web search and data mining. ACM, 2012.
25. Hu, Yu, Jiwoong Shin, and Zhulei Tang. "Incentive Problems in Performance-Based Online Advertising Pricing: Cost per Click vs. Cost per Action." Manag. Sci. (2015).
26. K. Ross, K. Fridgeirsdottir, Cost-Per-Impression and Cost-Per-Action Pricing in Display Advertising with Risk Preferences. Manuf. Serv. Oper. Manage. (2011)
27. C. Dellarocas, Double marginalization in performance-based advertising: Implications and solutions. Manag. Sci. **58**(6), 1178–1195 (2012)
28. Y.J. Hu, J. Shin, Z. Tang, *Pricing of online advertising: Cost-per-click-through Vs. cost-per-action. In System Sciences (HICSS), 2010 43rd Hawaii International Conference on (pp. 1-9), Hawaii IEEE* (2010)
29. N. Agarwal, S. Athey, D. Yang, Skewed bidding in pay-per-action auctions for online advertising. Am. Econ. Rev., 441–447 (2009)
30. "Proguard", [Online]. Available: http://developer.android.com/tools/help/proguard.html Accessed 29 Mar 2019.
31. "Java Decompiler", [Online]. Available: http://jd.benow.ca/ Accessed 29 Mar 2019.
32. "ipify", [Online]. Available: https://api.ipify.org/ Accessed 29 Mar 2019.
33. AdMob, "Monetize, Promote and Analyze your apps with AdMob" [Online]. Available: http://www.google.com/ads/admob/ Accessed 29 Mar 2019.
34. N. Kshetri, The economics of click fraud. IEEE. Secur. Priv. **8**(3), 45–53 (2010)
35. Faou, Matthieu, et al. "Follow the traffic: Stopping click fraud by disrupting the value chain." Privacy, Security and Trust (PST), 2016 14th Annual Conference on. IEEE, 2016.
36. V. Midha, The glitch in on-line advertising: a study of click fraud in pay-per-click advertising programs. Int. J. Electronic. Commerce. **13**(2), 91–112 (2008)
37. K. Asdemir, Ö. Yurtseven, M.A. Yahya, An economic model of click fraud in publisher networks. Int. J. Electron. Commerce. **13**(2), 61–90 (2008)
38. T. Dinev, Q. Hu, A. Yayla, Is there an on-line advertisers' dilemma? A study of click fraud in the pay-per-click model. Int. J. Electro. Commerce. **13**(2), 29–60 (2008)
39. D. Berrar, Learning from automatically labeled data: case study on click fraud prediction. Knowl. Inf. Syst. ***46***(2), 477–490 (2016)
40. X. Zhu, H. Tao, Z. Wu, J. Cao, K. Kalish, J. Kayne, in *Fraud Prevention in Online Digital Advertising*. Ad Fraud Categorization and Detection Methods (Springer International Publishing, 2017), pp. 25–38
41. Q. Zhang, W. Feng, in *Mathematical and Computational Approaches in Advancing Modern Science and Engineering*. Detecting Coalition Frauds in Online-Advertising (Springer International Publishing, 2016), pp. 595–605
42. M.S. Iqbal, M. Zulkernine, F. Jaafar, Y. Gu, in *High Assurance Systems Engineering (HASE), 2016 IEEE 17th International Symposium on (pp. 157-164). IEEE*. Fcfraud: Fighting click-fraud from the user side (2016)
43. X. Zhu, H. Tao, Z. Wu, J. Cao, K. Kalish, J. Kayne, *Fraud Prevention in Online Digital Advertising* (Springer, 2017)
44. W. Jianyu, W. Chunming, J. Shouling, G. Qinchen, L. Zhao, *Fraud Detection via Coding Nominal Attributes. In Proceedings of the 2017 2nd International Conference on Multimedia Systems and Signal Processing (pp. 42-45)* (ACM, 2017)
45. M.S. Iqbal, M. Zulkernine, F. Jaafar, Y. Gu, *Protecting Internet users from becoming victimized attackers of click-fraud* (Journal of Software: Evolution and Process, 2016)
46. G. Cho, J. Cho, Y. Song, D. Choi, H. Kim, Combating online fraud attacks in mobile-based advertising. EURASIP. J. Inf. Secur. **2016**(1), 2 (2016)
47. "Ad fraud loss slides to $6.5 billion worldwide", [Online]. Available: https://www.zdnet.com/article/ad-fraud-loss-plummets-to-6-5-billion-worldwide/ Accessed 29 Mar 2019.
48. Mouawi, R., Awad, M., Chehab, A., El Hajj, I. H., & Kayssi, A. (2018). Towards a Machine Learning Approach for Detecting Click Fraud in Mobile Advertizing. In 2018 International Conference on Innovations in Information Technology (IIT) (pp. 88-92). IEEE.
49. Wen, E., Cao, J., Shen, J., & Liu, X. (2018). Fraus: Launching Cost-efficient and Scalable Mobile Click Fraud Has Never Been So Easy. In 2018 IEEE Conference on Communications and Network Security (CNS) (pp. 1-9). IEEE.
50. M. Gabryel, *Data Analysis Algorithm for Click Fraud Recognition. In International Conference on Information and Software Technologies (pp. 437-446).* (Springer, Cham, 2018)
51. R. Kayalvizhi, K. Khattar, P. Mishra, *A Survey on Online Click Fraud Execution and Analysis In International Journal of Applied Engineering Research* (2018), pp. 13812–13816
52. Liang, Y. S., Chen, X., Chen, Y., & Xiao, P. (2018). Ad Fraud Under the Vertical Contract Structure. Xinlei and Chen, Yuxin and Xiao, Ping, Ad Fraud Under the Vertical Contract Structure (October 31, 2018).
53. M. Fallah, S. Zarifzadeh, Practical Detection of Click Spams Using Efficient Classification-Based Algorithms. Int. J. Inf. Commun. Technol. Res. **10**(2), 63–71 (2018)
54. Nagaraja, S., & Shah, R. (2019). Clicktok: Click Fraud Detection using Traffic Analysis. arXiv preprint arXiv:1903.00733.
55. N. Kshetri, S. Voas, *Online Advertising Fraud* (Published by the IEEE Computer Society, 2019)
56. P.S. Almeida, J.J. Gondim, Click Fraud Detection and Prevention System for Ad Networks. J. Inf. Secur. Cryptography. (Enigma) **5**(1), 27–39 (2019)
57. F. Dong, H. Wang, L. Li, Y. Guo, T.F. Bissyandé, T. Liu, et al., *Frauddroid: Automated ad fraud detection for android apps. In Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (pp. 257-268)* (ACM, 2018)
58. P. Dmytro, C. Oleg, in *Springer Nature Switzerland AG*. How Click-Fraud Shapes Traffic: A Case Study (2019)
59. C.M.R. Haider, A. Iqbal, A.H. Rahman, M.S. Rahman, An ensemble learning based approach for impression fraud detection in mobile advertising. J. Netw. Comput. Appl. **112**, 126–141 (2018)

Publisher's Note