AMERICAN UNIVERSITY OF BEIRUT

ASSESSMENT OF IN-APP ADS SECURITY VULNERABILITIES AND RESOURCE CONSUMPTION

by
RIWA HAIDAR MOUAWI

A thesis
submitted in partial fulfillment of the requirements
for the degree of Master of Engineering
to the Department of Electrical and Computer Engineering
of the Faculty of Engineering and Architecture
at the American University of Beirut

Beirut, Lebanon
April 2016

AMERICAN UNIVERSITY OF BEIRUT

ASSESSMENT OF IN-APP ADS SECURITY VULNERABILITIES AND RESOURCE CONSUMPTION

by

RIWA HAIDAR MOUAWI

Approved by:

Dr. Imad H. Elhajj, Associate Professor — Advisor
Electrical and Computer Engineering

Dr. Ayman Kayssi, Professor — Member of Committee
Electrical and Computer Engineering

Ali Chehab

Dr. Ali Chehab, Professor — Member of Committee
Electrical and Computer Engineering

Date of thesis defense: April 21, 2016

AMERICAN UNIVERSITY OF BEIRUT

THESIS, DISSERTATION, PROJECT RELEASE FORM

Student Name: ______Mouawi__________Riwa_________________Haidar________
Last First Middle

● Master's Thesis ○ Master's Project ○ Doctoral Dissertation

☑ I authorize the American University of Beirut to: (a) reproduce hard or electronic copies of my thesis, dissertation, or project; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes.

□ I authorize the American University of Beirut, **three years after the date of submitting my thesis, dissertation, or project,** to: (a) reproduce hard or electronic copies of it; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes.

Riwa Mouawi April 27, 2016

Signature Date

ACKNOWLEDGMENTS

I would like to thank first my advisor Prof. Imad ElHajj who has been my main guide throughout this thesis. I am really grateful for all the help and advice you have given me, for all the countless emails that you replied to, for all the countless meeting that we scheduled and for being such a talented professor and advisor.

I also like to thank my committee members Prof. Ali Chehab and Prof. Ayman Kayssi for their precious insights and help throughout this work.

I would like to thank my parents and sisters, your support was really a key factor for my success. As for my mother, a thank you is not enough to express how grateful I am for your support.

Finally, I want to thank my future husband Ayman Mukaddam; without his support and encouragement, I would not have finished this thesis. You encouraged me to do my Master's Degree at AUB from the beginning, and have always been there for me through thick and thin.

AN ABSTRACT OF THE THESIS OF

Riwa Haidar Mouawi for Master of Engineering
Major: Machine Intelligence

Title: Assessment of In-App Ads Security Vulnerabilities and Resource Consumption

Mobile advertising using in-app ads has increased in popularity along with the substantial number of current free mobile applications in the app stores. This relatively new type of advertising has raised several concerns during the past few years. The first concern is *resource consumption*, such as the battery consumption that mobile advertising entails and the network traffic overhead that it consumes to download the ads. The second concern is mobile ads *click fraud* that threatens the mobile advertising economy.

While several efforts revealed key observations regarding ads-related energy and bandwidth consumption, they did not compare these two types of consumptions among different ad networks. Unlike some previous work that just mentioned the ad networks associated with the tested apps, this thesis evaluates bandwidth and energy consumption and compares them among several popular ad networks that support ads for Android applications. The experimental procedure followed in this study demonstrated that resource consumption varies significantly among networks based on our statistical tests: For the same testing environment and duration, the bandwidth consumption in monetary value is *6.1$* for the ad network “Mobfox”, and *0.2$* for the ad network “Millennial Media”; the battery consumption in standby time is *1.3 min* for the ad network “AppFlood”, and *33.2 min* for the ad network “Flurry Mediation”. In addition, this study highlights a common behavior when fetching ads, where ads are fetched at the beginning of app runtime and displayed throughout the application session.

From a security perspective, although most of the 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. In this thesis, we propose a new crowdsource based system 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 data corresponding to different ad network-publisher-advertiser combinations, and a detailed view, where it is able to track the user’s engagement in each advertising website. Our 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 ads charging model* that benefits from our system to charge advertisers based on the duration spent in the advertiser’s website or any other measurable criteria.

CONTENTS

ILLUSTRATIONS

Figure Page

TABLES

CHAPTER 1

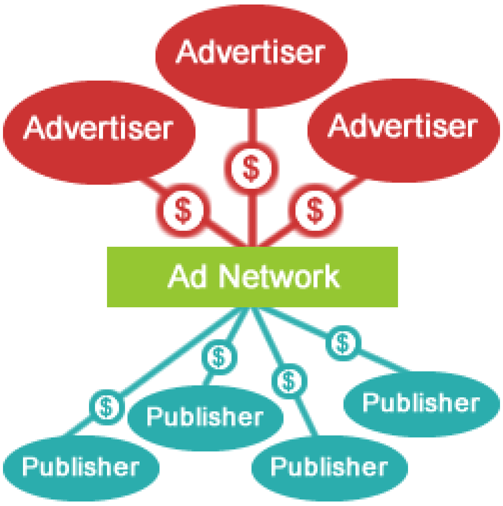
INTRODUCTION

This chapter presents an overview of mobile advertising, highlighting two major concerns in this industry: mobile ads resource consumption and mobile ads click fraud detection. In addition, it explains the different charging models adopted in mobile advertising. It also introduces the motivation and objectives of this thesis. Finally, this chapter presents an outline for the thesis.

1.1. Mobile Ads Overview:

With the increasing number of free apps in the app stores (as high as 84.8% 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 a report published by Juniper Research, it is predicted for in-app advertising to grow to around $17 billion by the year 2018 [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 them interchangeably in this thesis.



1.1-1: Mobile Advertising Main Components [67]

There are four main components in the mobile advertising community (figure 1.1-1):

1) *Advertiser*: An advertiser is any individual (not necessarily a technical expert), who is willing to pay to have his ads displayed.
2) *Publishers*: A publisher is any application's developer who wish to generate revenues by displaying ads in return in his/her application.
3) *Ad Network*: An ad network is a company specialized in mobile advertising who 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 follows an integration process (explained in section 3).
4) *User*: A user is any individual that uses a mobile application, and interacts with ads featured in this application (the interaction type is explained below).

1.2. Mobile Ads Resource Consumption

In-app ads are a great and easy way for the developer to achieve a large number of downloads and at the same time generate a profit. Nonetheless, this mobile advertising comes with a price. A recent study highlighted the fact that mobile ads cause a relatively significant network overhead that might in some cases cost more than the paid version of the app [3]. Another study showed that almost 65-70% of the total energy consumed is caused by ads in several free apps (such as Angry Birds and fchess) [4, 5].

1.2.1. Motivation and Objectives

Many efforts have been made to evaluate the power consumption caused by ads [6], and furthermore to mitigate this power consumption, such as adopting an ads prefetching technique [7, 8].

However, none of these studies compared this resource consumption among different ad networks. The first part of this thesis focuses on bandwidth and energy consumption and their comparisons among several popular ad networks.

1.3. Mobile Ads Click Fraud Detection

Mobile ads click fraud is another major concern in the advertising community. Click fraud, also known as click spam, is when the user clicks on the ad in a mobile application not because of interest in this 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 [36]. Researchers estimated advertisers' loss caused by click fraud at $1 billion in 2013 [35]. In fact, it is predicted for advertisers to lose **$ 6.3 billion** due to click fraud in 2016 [68].

Furthermore, in [37], 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.

1.3.1. Motivation and Objectives

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 for the client from a potential conspiracy between publishers and ad networks. In addition, ad networks are not able to monitor the user's activity, once redirected to the advertising site after clicking the ad. 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 [38-44] and proposed new click fraud detection systems for ad networks without collaborating with advertisers [45-49]. Other studies proposed new click fraud detection systems that enable advertisers to detect click fraud without collaborating with ad networks [36, 50, 51]. Nonetheless, these advertisers click fraud system are prone to a high false positive rate since they do not offer a global view (many clicks from one application), but instead judge click fraud on a per click basis.

In the second part of this thesis, we propose a new click fraud system that follows 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 crowd source 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.

1.4. Mobile Ads Charging Models

There are three main mobile charging models used nowadays in the market known as: 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 from one ad network to another (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 "C*onversion".* "C*onversion*" 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) [66].

In the cost-per-thousand-impressions (**CPM**) charging model, the conversion is an ad view: The advertiser pays for each 1,000 ad displays; the publisher is paid in return a percentage of the ad network's profit, whenever a user views an ad in his 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) charging model, the conversion is an action: The advertiser pays for each performed action. Once the user clicks on a mobile ad, he will be redirected to the advertiser's website where he might complete an action. An action can be a simple registration by the user, a purchase, a file download… 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.

1.4.1. Motivation and Objectives

The CPC model and CPM models are usually more desirable by publishers [65] 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 ads-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 reach and is based on the quality of the advertiser's website.
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 won't complete any of the actions determined by current CPA models (such as registration, download, purchase…). In fact, in [66] 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 cosmetics products click on a cosmetic ad and browse the advertiser's website without completing any purchases. The next week, after she had become aware of this

website, she browses the cosmetics website and completes 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.

Therefore, we propose a *new mobile ads charging model* that benefits from our CFC system to charge advertisers based on the duration spent in the advertiser's website or any other measurable criteria.

1.5. Thesis Structure

The remainder of this thesis is organized as follows: Chapter 2 discusses first, the related work highlighting the resource consumption caused by ads followed by the related work highlighting the click fraud detection systems proposed in the literature and finally, the related work of the CPA domain. Chapter 3 presents the first part of this thesis: *A comparison of in-app ads traffic in different ad networks from a resource consumption's perspective*. It explains the followed experimental procedure, the results of our test implementation, the analysis of the results, and a conclusion of this first part. Chapter 4 presents the second part of this thesis: *Using crowdsourcing for click fraud* *detection*. It presents the architecture of our proposed system, the experimental setup, the results of our implementation, the analysis of the obtained results, the new proposed CPA model, and a conclusion of this second part. Chapter 5 presents the conclusion of this thesis and the future work.

CHAPTER 2

LITERATURE REVIEW

This chapter provides first a literature review of the different resource consumption studies related to mobile ads, second a literature review of the different click fraud detection approaches proposed in the literature and third a literature review of the different charging models studies related to CPA.

2.1. Resource Consumption Literature

In recent studies, three approaches have been followed in terms of assessing the effect of in-app ads on the mobile device resources consumption.

2.1.1. Bandwidth Consumption

The first approach evaluates the bandwidth consumption caused by mobile ads: Zhang et al. analyzed the network overhead caused by ads and analytics data in applications and assessed how much this overhead costs in terms of bandwidth consumption and the associated monetary value. They showed that in most cases the free version of an app actually costs more than the paid version due to ads-related bandwidth consumption [3]. However, they did not study energy consumption caused by ads.

2.1.2. Battery Consumption

The second approach evaluates the battery consumption caused by in-app ads: Pathak et al. proposed an energy profiler tool called "eProf" where energy is evaluated per thread and per process in each application, taking into consideration the asynchronous power behavior found in smartphones, where “the effect on a component’s power state due to a program entity lasts beyond the end of that program entity” [4]. According to their study, almost 65-70% of the total energy consumed is caused by ads in several free apps (such as Angry Birds and fchess) [4,5].

Prochkova et al. compared energy consumption between mobile game apps that contain ads and mobile games that don't. They demonstrated how the increasing ad refresh rate directly and negatively affects power consumption [6].

Gui et al [32] investigated both bandwidth and battery consumption caused by ads in 21 different applications. By comparing these applications with and without ads, they concluded that with the use of ads, the bandwidth consumption increases to 97% more and the battery consumption increases to 15% more. However, their study was based on only one ad network (Google Mobile Ads [15]).

2.1.3. Resource Consumption Minimization

The third approach proposes a solution for minimizing resource consumption: Vallina-Rodriguez et al. used a large data set of traffic from a European mobile carrier to evaluate network ads traffic and highlighted several inefficiencies in the current ad systems, such as redundant downloaded data. Based on these inefficiencies, they proposed a new technique of prefetching ads called "adCache” that reduces resource consumption [7].

Chen et al. also assessed the benefits of prefetching ads. The results showed that ads behavior changes between different apps, which makes it hard to efficiently save energy using this prefetching technique [8].

P. Mohan et al. contributed to the ads prefetching technique by proposing a model that takes into consideration usage prediction such as the number of ad slots needed for future use depending on the user's behavior, while also respecting the ad expiration deadline and other constraints [9].

Khan et al. presented "CAMEO" a new mobile advertisements framework that uses predictive prefetching of ads to reduce resource consumption. In addition, CAMEO's main goal is to establish a negotiation protocol between ad networks and ISP to reduce the bandwidth cost imposed by the ISP [10, 11].

Seneviratne et al. addressed user privacy issues in terms of targeted ads in mobile applications. They proposed a new framework called “CAARETA” based on personal cloudlets that works as an intermediate agent between the ad network and the mobile device. They explained how this framework can respect the user's privacy while achieving targeted ads and at the same time helps reduce resource consumption. However, they did not perform any implementation [12].

2.1.4. Resource Consumption Literature Summary

While these studies revealed several key observations regarding ad-related energy consumption, they did not compare resource consumption among different ad networks. In addition, some of these works evaluate only the network overhead associated with ads without focusing on the related battery consumption [3], whereas others focus only on battery consumption caused by ads, without highlighting the related network traffic generated [6]. Our work evaluates both the bandwidth and the energy consumption of ads and compares them among several chosen ad Networks.

2.1.4-1: Key Features of different ads-related analysis approaches.

Ref.	Focuses on energy consumption	Focuses on network overhead	Solution for minimizing resource consumption	Compares traffic and power consumption
[6, 8]	Yes	No	No	No
[3]	No	Yes	No	No
[12]	No	Yes	Yes	No
[9]	Yes	No	Yes	No
[4, 5, 7, 10, 11]	Yes	Yes	Yes	No
Our Work	Yes	Yes	No	**Yes**

Table 2.1.4-1 summarizes the key features offered by the related work to mobile ads resource consumption and our work. Although our approach doesn't offer any techniques to decrease the power consumption of ads, it does offer a new perspective as to how ad networks behave differently and thus vary in their associated ads resource consumption.

2.2. Click Fraud Detection Literature

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

2.2.1. 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 any 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 one second* on their landing page [36].

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 [38].

Following a machine learning approach while testing over 165K 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 [39].

Blizzard T. 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) [40].

Alrwais S. 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 ads impressions and ad clicks from victim publishers [41].

Stone-Gross B. 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 [42].

Pearce P. 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 100K per day [43].

Miller B. et al. operated two families of bots in a controlled environment to monitor how botnet ad fraud, also known as clickbot, works in action [44].

2.2.2. Click Fraud Detection - Ad Network's Side

The second category in the literature proposes solutions that perform 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 [45]. Juels et al. proposed a new ad fraud mitigation technique that protects the pay-per-click charging model by cryptographically authenticating legitimate users [46]. Vasumati D. et al. used a data mining classification algorithm to identify fraudulent clicks [47]. Li W. et al. proposed a new ad framework that uses ARM Trustzone services to securely fetch ad placement and application-related information, and then to sign this information with the user's signature [48]. Haddadi H. 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 her [49].

2.2.3. Click Fraud Detection - Advertiser's Side

The third category of literature proposes solutions that perform 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 criterion such as the duration spent by the user on their websites [36]. Xu H. 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 [50]. Vani M. 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, time of access, etc.) and employed a node-tag based algorithm to differentiate spam and non-spam applications [51].

2.2.4. Click Fraud Detection Literature Summary

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 proposed fraud detection 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 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 2.2.4-2 summarizes the key features in the click fraud literature and compares them to our work (CFC). Although our approach doesn't investigate existing click fraud attacks and threats, it offers a solution that protects both advertisers and ad networks.

2.2.4-2: 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
[38 - 44]	No	No	Yes	No
[36]	No	Yes	Yes	No
[45 - 49]	Yes	No	No	No
[50, 51]	No	Yes	No	No
CFC	Yes	Yes	No	**Yes**

2.3. 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.

2.3.1. CPA – General Overview

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

Pechuán et al [57] presented an overview of the CPA model, its advantages and disadvantages. Mahdian et al [58] also explained how the CPA model works: its framework, its advantages, and the challenges that it faces (in terms of feasibility, user privacy….). Rosales et al [59] Provided an analysis of conversion rates in CPA or CPC models (where conversion is not guaranteed as opposed to CPM). By analyzing ads 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 affects CVR but doesn't affect the CTR.

Many studies addressed the CPA charging model from an economical approach: Hu et al [60] 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 [61] 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 [62] also explained several economical concerns of the CPA model. Hu et al [63] 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.

2.3.2. 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 [57] 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 proposed possible solutions/improvements for CPA scam detection, however, they did not present any detail explanations to support their proposed solutions. A major concern in the CPA security field is the possible misreports of actions by advertisers: Agarwal et al [64] 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 [58] 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 [65] presented a mathematical bidding representation that can be used in the CPA pricing model. Similarly to [58], they demonstrated how their approach limits the incentive of advertisers being dishonest when reporting actions.

Ding et al [66] proposed a simple solution for detecting when advertisers 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 advertisers' websites after clicking on their ads.

2.3.3. CPA Charging Model Literature Summary

While the *rank-by-revenue* bidding model potentially reduces the advertisers' incentive of under-reporting performed actions on their websites, 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 doesn't rely on the advertiser's input is required. Furthermore, as explained earlier, the actions used in the current CPA model might not reflect true user's interests in many cases, thus we propose a *new mobile ads charging model* that benefits from our CFC system to charge advertisers based on the *duration* spent in the advertiser's website (or any other measurable criteria). Table 2.3.3-3 compare our proposed CPA model with the literature.

2.3.3-3: CPA Literature vs Our Work

Reference	Present CPA overview (not security)	Present CPA overview (security)	Present Solution controlled by advertiser	Present Solution controlled by party trusted by advertiser and publisher
[59 - 63]	Yes	No	No	No
[57, 58]	Yes	Yes	Yes	No
[64]	No	Yes	No	No
[65, 66]	No	Yes	Yes	No
Our Work	Yes	No	No	Yes

2.4. Summary

This chapter presented the literature review of two major concerns in the mobile advertising industry: Resource consumption and click fraud. In addition, it presented the related work performed in the CPA charging model. Furthermore, It introduced the contribution performed in this thesis to overcome these concerns.

CHAPTER 3

RESOURCE CONSUMPTION COMPARISON

This chapter presents the first part of the thesis: *a comparison of in-app ads traffic in different ad networks*. It explains the followed experimental procedure, the results of our implementation, the analysis of the obtained results and a conclusion of this first part.

3.1. Implementation

To study the effect of in-app ads on bandwidth and battery consumption, the following experimental procedure was followed. To integrate ads in an application such as a game, the developer has a wide range of ad networks to choose from. In fact, according to a recent study, there are currently over 400 mobile ad networks [13].

Although several ad types are used such as banners, custom native ads, notifications, interstitial ads (an image that covers the entire screen), video ads, among many other types, we evaluate the following ad types:

3.1.1. Banner ads

Banner ads are in fact considered to be the most popular type of mobile ads [14]. Therefore, we tested ten different ad networks for this type of ads: AdMob [15], Flurry [16], Appia [17], AppFlood [18], RevMob [19], MobPartner [20], Millennial Media [21], InMobi [22], MoPub [23] and MobFox [24]. In addition, we tested two Mediation ad networks where each hosts ads come from multiple sources (ad networks): AdMob Mediation [15] and Flurry Mediation [16].

For each selected ad network, an Android application was created. Each application displays a white background and a banner (with dimensions 350 by 50 pixels) at the top of the screen. At run time, the application connects to the hosted ad network, fetches the ads and displays them in the banner.

3.1.2. Interstitial ads

Interstitial ads are most effective while promoting games [25]. We tested ten different ad networks for this type of ads: Millennial Media [21], InMobi [22], MoPub [23], MobFox [24], Flurry [16], AppFlood [18], Appia [17], LeadBolt [26], MobPartner [20] and RevMob [19].

For each selected network, an Android application was created. Each application displays a white background. At launch time the application connects to the hosted ad network, fetches the ads and displays a static full screen interstitial ad.

3.1.3. Video Interstitial Ads

With high user views, video ads have gained popularity among developers [25]. We tested five different ad networks for this type of ads: Millennial Media [21], InMobi [22], MobFox [24], MoPub [23], and Flurry [16].

Also, in this case, for each selected network, an Android application was created. Each application displays a white background. At launch time the application connects to the hosted ad network, fetches the ads and displays a full screen video interstitial ad.

3.1.4. Implementation Settings

The created applications connect to the Internet using Wi-Fi only (Mobile data option is turned off). The same level of brightness for device screen (approximately 45%) and same screen is used during this implementation.

The application "Power Tutor" [4] is used to obtain the battery consumption values. The "Data usage" option integrated by default in the Android operating system is used to get the bandwidth consumption values. The experiments were performed on a "S4 Mini: GT-I9190" smartphone running Android version 4.2.2.

3.1.5. Implementation Challenges

Several ad networks (Airpush [27], Admoda [28], Smaato [29], GreyStripe [30], madvertise [31]) did not approve our request to test their ads for this educational purpose.

In addition, several ad networks (InMobi [22], Millennial Media [21], MoPub [23], MobFox [24] and MobPartner [20]), did not successfully fetch any ads from Lebanon, this is due to the fact that these ad networks do not currently have any ads targeting Lebanon.

Therefore, the experiments were conducted from Lebanon and then repeated when connected to a router with VPN connection to Canada which made our device appear as if it is in Canada. This provided insight into whether location plays a role in the resource consumption. Note that the focus of the study is a comparative analysis and thus the accuracy of the absolute values obtained is not as critical to the analysis as is the relative differences. Therefore, the main experimental focus was on the consistency of the setup and tests conducted.

3.1.6. Ads Integration Steps

Most of the ad networks follow the same integration steps: First, the developer must create an account, an application, and an "ad placement". Also, the developer must download and integrate the ad network's SDK and finally integrate in the application a custom component that either represents the banner or displays a full screen image/video.

3.2. Implementation Results

Figure 3.2-2 and figure 3.2-3 show the results of the banner ad testing experiments performed in this study. AdMob [15], Flurry [16], AdMob Mediation [15], Flurry Mediation [16], and MobFox [24] enable the developer to control the refresh interval of the ad, meaning how often the ads change. This value typically ranges from 30 seconds to 120 seconds. We tested these ad networks for two refresh intervals: 30 seconds and 60 seconds. Appia [17], AppFlood [18], MobPartner [20], Millennial Media [21], InMobi [22], MoPub [23] and RevMob [19] follow a different refresh mechanism: instead of refreshing the ad according to a defined refresh interval, the ad is refreshed whenever the application is first launched and remains the same until the application is re-launched again. Therefore, we tested these seven ad networks for one app launch.

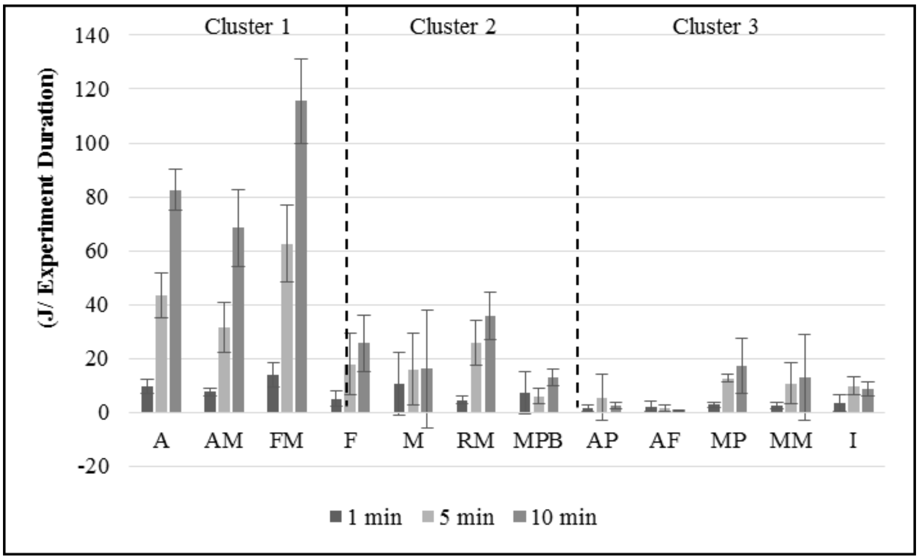
We used the following abbreviation for the ad networks' names: Admob: A; Flurry: F; Admob Mediation: AM; Flurry Mediation: FM; Mobfox: M; Appia: AP; Appflood: AF; Revmob: RM; Mobpartner: MP; Millennial Media: MM; Inmobi: I; Mopub: MPB; Leadbolt: L.

The results in figures 3.2.2-7 are sorted by clusters that we performed using Matlab. The clustering process and objective will be discussed later in this section.

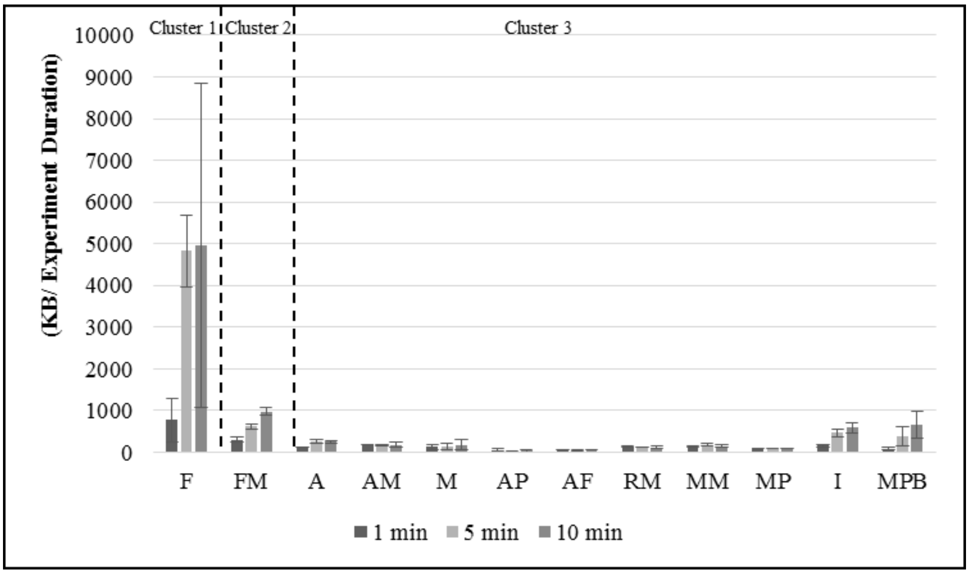
For each refresh criterion (or app launch in the case of the second set of ad networks), we tested three experiment durations: 1 minute, 5 minutes and 10 minutes. The experiment was repeated five times, in order to obtain more reliable statistics. Then, the averages and standard deviations for these five experiments were calculated (as shown in figures 3.2.2-7).

Figure 3.2-4 and figure 3.2-5 show the results of the interstitial ads experiments performed in this research. We tested each ad network for one-minute experiment duration five times each.

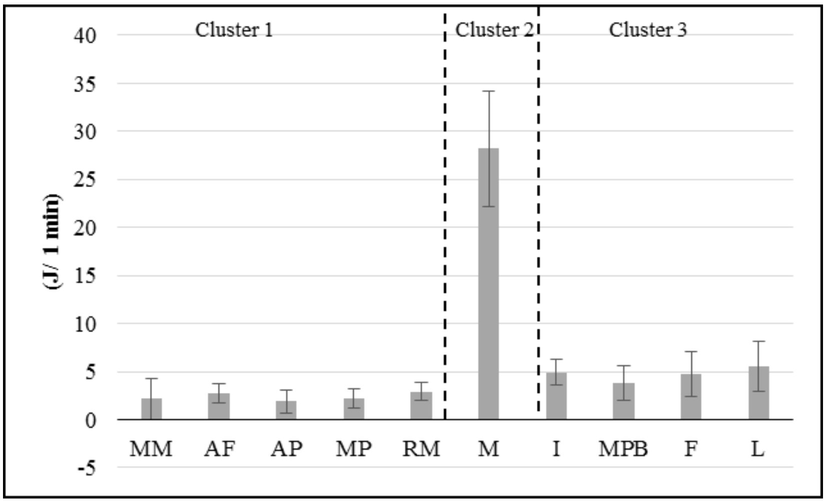
Figure 3.2-6 and figure 3.2-7 show the results of the video interstitial ads experiments performed in this research. We tested each ad network for one- and five-minute experiment durations, five times each.



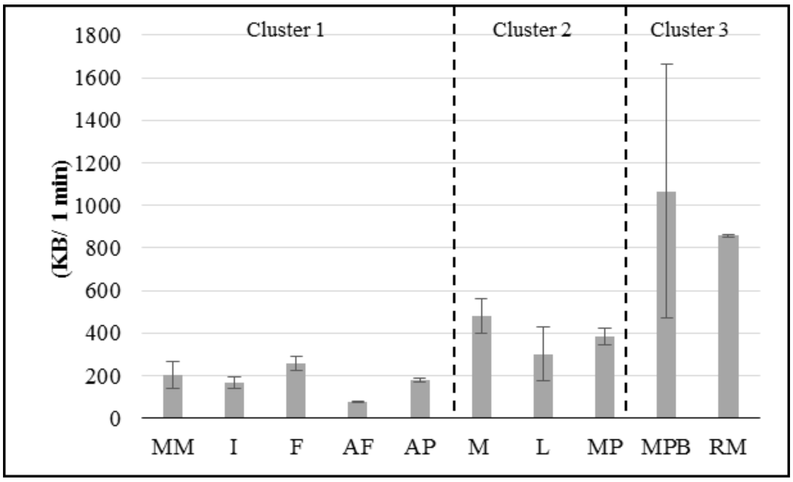
3.2-2 : Banner Ads Battery Results



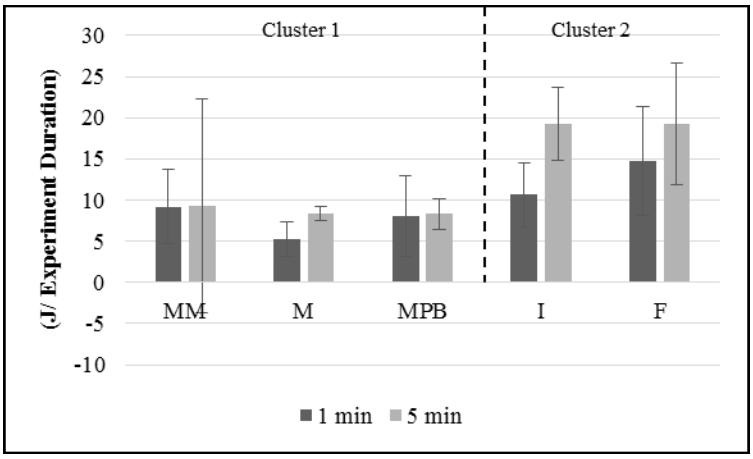
3.2-3: Banner Ads Bandwidth Results



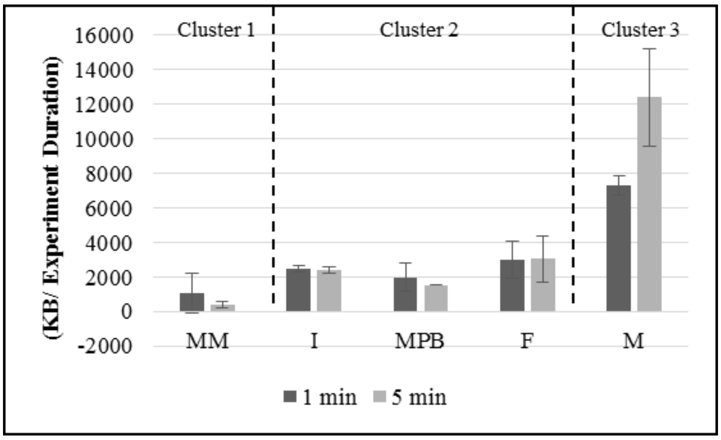
3.2-4: Interstitial Ads Battery Results



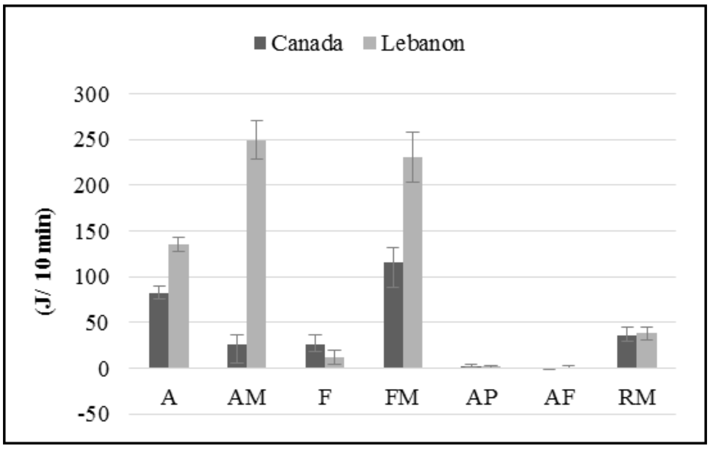
3.2-5: Interstitial Ads Bandwidth Results



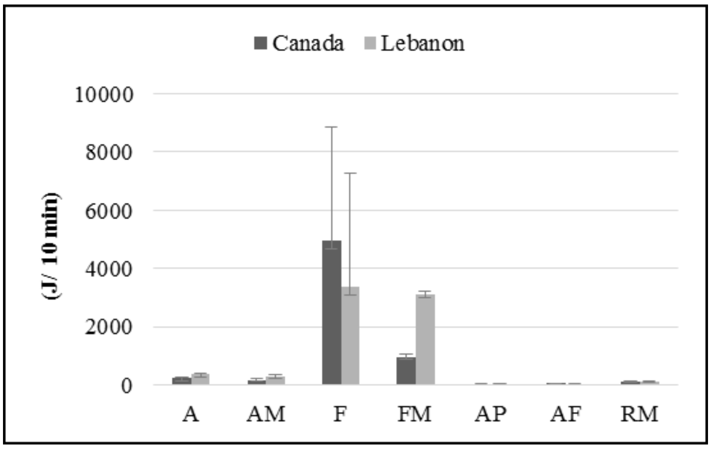
3.2-6: Video Ads Battery Results



3.2-7: Video Ads Bandwidth Results



3.2-8: Banner Battery Results in two locations



3.2-9: Banner Bandwidth Results in two locations

Based on the information collected during our experiments (summarized in the figures), we used Matlab to do a K-means clustering of the bandwidth and battery consumption for each ads type. By clustering our results (shown in table 3.2-3) we are able to categorize the ad networks into three clusters (for each ad type and measurement type). This categorization helps us better assess the difference between the ad networks in resource consumption. K-means clustering enables us to specify the number of clusters that we wish to have; therefore, we chose to have three clusters (high, medium, low) for each experiment (except for the video ads battery results where two clusters were used, since the video battery results clearly seemed to belong to two groups only).

3.2-4: K-means Clustering Results and Associated t-test Results

Ad Type	Measurement Type	Cluster 1	Cluster 2	Cluster 3	P value (C1, C2)	P value (C1, C3)	P value (C2, C3)
Banner	Battery (J/ 10 min)	A; AM; FM	F; M; RM; MPB	AP; AF; MP; MM; I	2.80243E-12	3.29468E-28	1.61078E-05
	Bandwidth (J/ 10 min)	F	FM	A; AM; M; AP; AF; RM; MM; MP; I; MPB	2.2302E-08	5.67053E-10	5.48044E-07
Interstitial	Battery (J/ 1 min)	MM; AF; AP; MP; RM	M	I; MPB; F; L	0.000302637	0.00066791	0.000347037
	Bandwidth (J/ 1 min)	MM; I; F; AF; AP	M; L; MP	MPB; RM	8.43095E-07	9.71991E-05	0.000822249
Video	Battery (J/ 1 min)	MM; M; MPB	I; F	NA	0.026425576	NA	NA
	Bandwidth (J/ 1 min)	MM	I, MPB, F	M	0.010361437	0.00020116	0.001157685

3.2-5: Canada and Lebanon t-test Results

	P value (Canada - Lebanon)
Battery	0.127714802
Bandwidth	0.450460748

To evaluate the significance of the bandwidth and battery consumption differences between the different clustered ad networks, we applied the statistical test known as t-test and calculated the resulting P-value between the clusters for each ad type (banner, interstitial and video) and each measurement type (bandwidth and battery). As shown in Table 3.2-4, all the P-values are below 0.05 (5% error), which means that for a given ad type, the battery consumption between the different ad networks clusters is statistically significantly different, and the bandwidth consumption between the different ad networks clusters is also statistically significantly different.

By calculating the average bandwidth/battery of each cluster A, compared to cluster B, for each ad type, we got the following results:

For the banner battery, cluster 1 is 2.93 times more than cluster 2, and 9.57 times more than cluster 3; cluster 2 is 1.69 times more than cluster 3. For the banner bandwidth, cluster 1 is 4.09 times more than cluster 2, and 20.27 times more than cluster 3; cluster 2 is 3.18 times more than cluster 3.

For the interstitial battery, cluster 2 is 10.93 times more than cluster 1, and 4.94 times more than cluster 3; cluster 3 is 0.5 times more than cluster 1. For the interstitial bandwidth, cluster 3 is 4.44 times more than cluster 1, and 1.48 times more than cluster 2; cluster 2 is 1.19 times more than cluster 1.

For the video battery, cluster 2 is 0.7 times more than cluster 1. For the video bandwidth, cluster 3 is 6.09 times more than cluster 1, and 1.95 times more than cluster 2; cluster 2 is 1.39 times more than cluster 1.

Based on the data collected during our experiments (summarized in figure 3.2-8 and figure 3.2-9), we applied the statistical t-test (as shown in table 3.2-5) and calculated first, the resulting P-value between the bandwidth consumption of the different ad networks in Lebanon and the bandwidth consumption of the different ad networks in Canada. Then we calculated the resulting P-value between the battery consumption of the different ad networks in Lebanon and the battery consumption of the different ad networks in Canada. Both P-values are above 5%, which means that the bandwidth and battery consumption do not differ statistically significantly between the two locations (Lebanon and Canada).

Based on recorded standby energy values for several time intervals in [33], we used linear regression to predict the equivalent standby time lost due to ad network battery consumption as shown in Table 3.2-6. For example, by using an application that displays AdMob banner for only 10 minutes, we will be losing 24.8 minutes of standby energy.

Based on the average cost per MB of AT&T in 2015 [34] ($0.07 per MB), we calculated the equivalent monetary value of our recorded results for a one week of use (as shown in table 3.2-7). For example, using an application that hosts Flurry banner ads for a 10 minutes duration (for a period of one week) costs 2.5$, whereas using an application that hosts Mobfox video ads for a 5 minutes duration (for a period of one week) costs 6.1$.

3.2-6: Battery Consumption Results in Standby time (min)

Ad Type	**Banner**	**Interstitial**	**Video**
Experiment Duration	**10 min**	**3 min**	**5 min**
A	*24.8*	NA	NA
F	10.7	2.9	4.7
AM	19.2	NA	NA
FM	**33.2**	NA	NA
M	13.9	**23.8**	0.4
AP	1.8	0.4	NA
AF	1.3	1.2	NA
RM	11.8	1.3	NA
MP	5.7	0.7	NA
MM	4.6	0.7	4
I	2.7	3.1	2.3
MPB	8.2	2.1	2.7
L	NA	3.7	NA

3.2-7: Bandwidth Results in Monetary Values ($) (for one week)

Ad Type	**Banner**	**Interstitial**	**Video**
Experiment Duration	**10 min**	**1 min**	**5 min**
A	0.2	NA	NA
F	2.5	0.2	1.5
AM	0.1	NA	NA
FM	0.5	NA	NA
M	0.1	0.3	***6.1***
AP	0.1	0.1	NA
AF	0.1	0.1	NA
RM	0.1	0.5	NA
MP	0.1	0.2	NA
MM	0.1	0.2	0.2
I	0.3	0.1	1.2
MPB	0.4	0.6	0.8
L	NA	0.2	NA

3.3. Results Analysis

The results shown in the figures highlight several key findings related to in-app ads.

1) While battery consumption has increased with the use of mediation in Flurry ad network, it has decreased with the use of mediation in adMob ad network (shown in fig. 3.2-2).
2) On the other hand, bandwidth consumption has actually decreased with the use of mediation for Flurry ad network and remained the same for adMob (shown in fig. 3.2-3).
3) The bandwidth and the battery consumption are not always directly related: For example, in comparison to other ad networks, AdMob has a high battery consumption level with a low bandwidth consumption level, whereas Flurry has a low battery consumption level with a high bandwidth consumption level (shown in fig. 3.2.2-7).

4) Most of the experiments showed that the battery consumption is not consistent across the experiment's duration (shown in fig. 3.2-2, 3.2-4, 3.2-6, and 3.2-8): it is higher when the application first launches in the [0-1] minute interval than in the rest of the experiment's duration. For example, if we use an application that displays Millennial Media video ads for a 5 minutes interval, we will have a battery consumption of 9.32 J. However, if we launched the same application 5 times (for a one minute interval for each time) we would have a 5 * [1 minute battery consumption= 9.22] = 5 × 9.22 = 46.1 J.

5) In most of the experiments, the bandwidth consumption is not consistent across experiment's duration (shown in fig. 3.2-3, 3.2-5, 3.2-7, and 3.2-9): it is much higher when the application first launches [0-1] minute interval and it decreases with duration. For example, in a one minute interval using Flurry video ads, the bandwidth consumption is **3010 KB**, whereas, in a 5 minutes interval it is **3034 KB**.

6) Using the application "Shark for root" on a rooted device (Samsung S3 GT-I9300 running Android version 4.4.4), we noticed that there exists an authentication process between the application and the ad network's server at the beginning of the application's session. We believe this authentication process causes an inconsistency in both bandwidth and battery consumption throughout the application's session (demonstrated in points 4 and 5).

7) The battery consumption of ad networks varies with the use of different ad categories (shown in fig. 3.2-2, 3.2-4 and 3.2-6). For example, for a one

minute interval, the battery consumption using InMobi is 3.38 J for banner ads, **4.92 J** for interstitial ads and **10.64 J** for video ads. In fact, by comparing the battery consumption of the same ad network, for the same duration, with the use of different ad categories (banner, interstitial and video ads), we found that interstitial ads have the lowest battery consumption, while video and banner ads are very similar.

8) Expectedly, the bandwidth consumption of the same ad network is very different with the use of different ad categories (shown in fig. 3.2-3, 3.2-5 and 3.2-7). For example, for a one minute interval, it is **1067 KB** for MoPub interstitial ads and **87.49 KB** for MoPub banner ads, whereas it is **1968 KB** for MoPub video ads.
9) Based on Table 3.2-5 P-value results (above 5%), we can conclude that the bandwidth and battery consumption do not differ significantly between two locations (Lebanon and Canada).
10) As shown in the t-test results in Table 3.2-4, for a given ad type, the battery consumption between the different ad networks clusters is statistically significantly different, and the bandwidth consumption between the different ad networks clusters is also statistically significantly different.

3.4. Resource Consumption Analysis Conclusion and Future Work

This first part of the thesis evaluates the bandwidth and battery consumption of in-app ads among several ad networks. The experimental procedure followed in this study demonstrates that resource consumption varies among networks and is usually consistent in the same ad network for a given duration. In addition, this study highlights a common behavior of fetching ads between several ad networks where ads are fetched at the beginning of runtime and displayed throughout the application session. This variation and significance of resource consumption indicates that developers need to take into consideration the bandwidth and battery consumptions when choosing the right ad network. On the other hand, the ad networks should adopt an ad-fetching mechanism that does not include large network overhead or unnecessarily heavy battery consumption. Furthermore, ad networks should adopt a transparent approach where they specify clearly the bandwidth and battery consumption associated with their ads fetching platform. Future work includes the evaluation of additional ad networks, testing on several devices, and a more accurate way to measure the variation of this consumption across time.

3.5. Summary

This chapter presented the first section of this thesis: a comparison of mobile ads traffic between different ad networks in terms of bandwidth and battery consumption. It explained the experimental procedure that was followed in this section, the obtained results, and its analysis. In addition, it provides an insight on the future work that could enhance this study and a conclusion of this section.

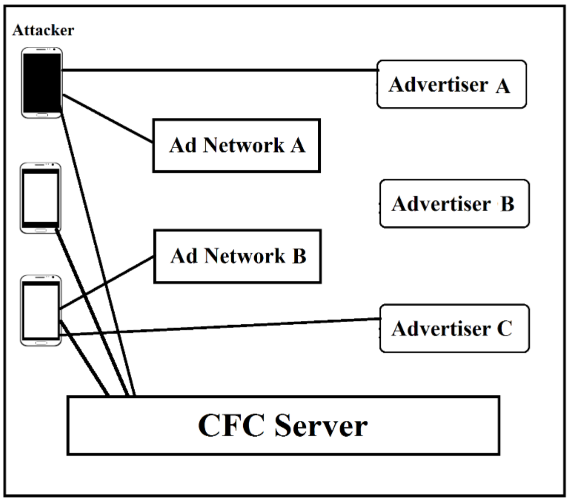
CHAPTER 4

CLICK FRAUD DETECTION

This chapter presents the second part of this thesis: click fraud detection using a crowdsourcing approach. It presents the architecture of our proposed system, the experimental setup, the results of our implementation, the analysis of the obtained results, and a conclusion of this part of the thesis.

4.1. Proposed Solution

Our proposed system is composed of four components (as shown in Figure 4.1-10).



4.1-10: CFC Proposed solution

4.1.1. Phone Component

The phone component is on the client side, and similar to traditional ad network frameworks, the publisher integrates in her app an ad component in the form of a single jar file, which is published by the ad network on its website after merging it with another jar file created by the CFC party in order to handle click fraud detection. The new file has its own billing mechanism that handles the selection of ads to show. This phone component fetches ads, displays them, and manages clicks by displaying the advertised websites on ad clicks and sending clicks information to the CFC party simultaneously (explained in section 4.2).

4.1.2. Ad Network Component

This component behaves similarly to classical ad network systems; it 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.

4.1.3. Advertiser Component

After a user clicks on an ad featured by the phone component, she will be redirected to the advertiser's website, which represents the advertiser component.

4.1.4. CFC Component (Server Side)

After collecting a large number of clicks (to be determined by the CFC administrator) from different phone components of ad networks using the CFC service, a crowdsource based calculation is performed. We will refer to this click-fraud-crowdsourcing-algorithm as "*CFCA*" in the rest of the thesis. The motivation behind our crowdsource based approach is based on two main needs: on 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), however, the judgment is per click and is therefore prone to a high false positive rate [36].

Accordingly, if a user clicks on an ad, and after being redirected to the advertiser's website, she shows no interest in this website, she will be flagged as malicious for exiting the website after few seconds. 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.

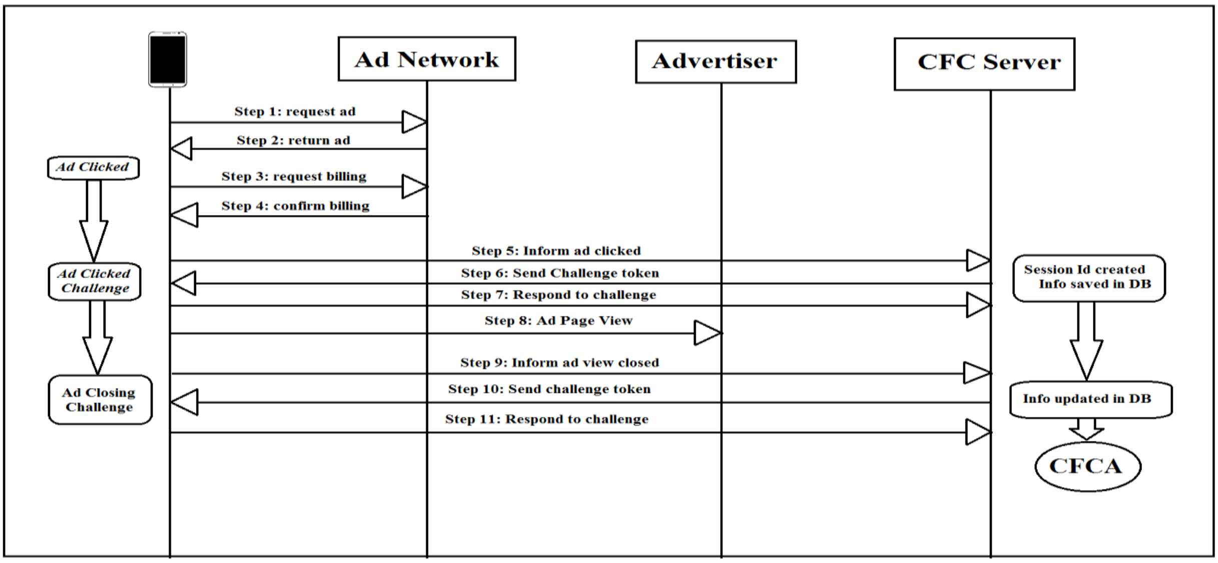
In addition to the CFCA, the CFC party manages several APIs that communicate with the CFC phone component integrated in the client side and a corresponding online database (explained in section 4.2).

4.1.5. CPA Enhanced Model

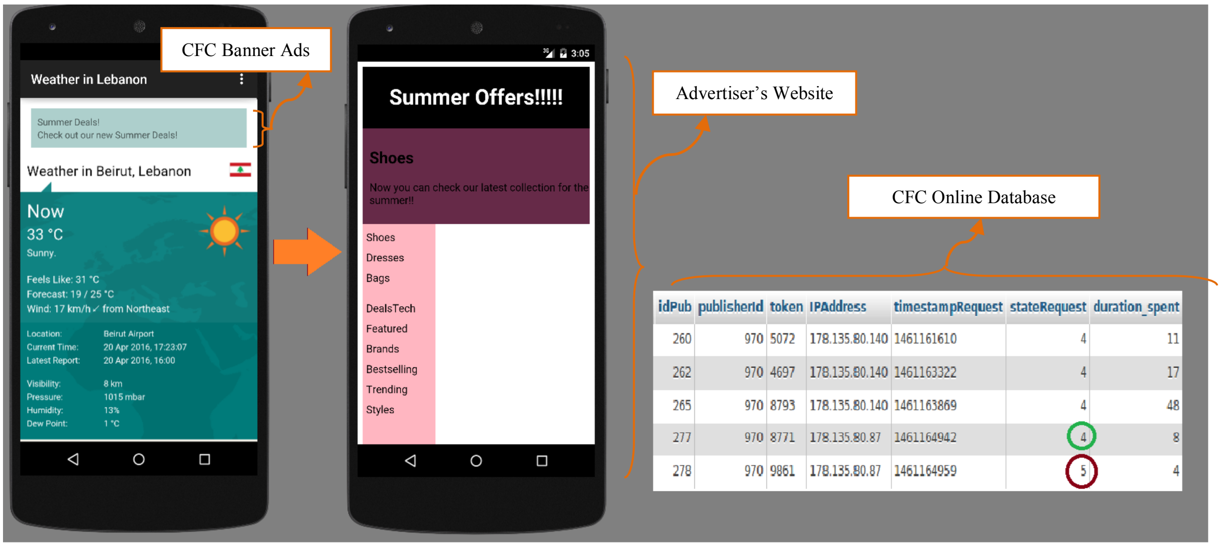
Our proposed mobile ads charging model benefits from our CFC system to charge advertisers based on different measurable criteria such as the duration spent in the advertiser's website. As opposed to current CPA systems that relies on the advertisers' to report the actions to be charged for, our CFC model 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. Our system in fact is transparent to both advertiser and publisher.

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 one minute.

4.2. Phone Component CFC Steps


Ad Network
Advertiser
CFC Server
Step 1: request ad
Step 2: return ad
Ad Clicked
Step 3: request billing
Step 4: confirm billing
Step 5: Inform ad clicked
Step 6: Send Challenge token
Step 7: Respond to challenge
Step 8: Ad Page View
Ad Clicked Challenge
Session Id created Info saved in DB
Step 9: Inform ad view closed
Step 10: Send challenge token
Step 11: Respond to challenge
Ad Closing Challenge
Info updated in DB
CFCA

4.2-11: CFC Steps



Weather in Lebanon

Summer Deals!
Check out our new Summer Deals!

Weather in Beirut, Lebanon

Now
33 °C
Sunny.

Feels Like: 31 °C
Forecast: 19 / 25 °C
Wind: 17 km/h ↙ from Northeast

Location: Beirut Airport
Current Time: 20 Apr 2016, 17:23:07
Latest Report: 20 Apr 2016, 16:00

Visibility: 8 km
Pressure: 1015 mbar
Humidity: 13%
Dew Point: 1 °C

3:05

Summer Offers!!!!!

Shoes

Now you can check our latest collection for the summer!!

Shoes
Dresses
Bags

DealsTech
Featured
Brands
Bestselling
Trending
Styles

idPub	publisherId	token	IPAddress	timestampRequest	stateRequest	duration_spent
260	970	5072	178.135.80.140	1461161610	4	11
262	970	4697	178.135.80.140	1461163322	4	17
265	970	8793	178.135.80.140	1461163869	4	48
277	970	8771	178.135.80.87	1461164942	4	8
278	970	9861	178.135.80.87	1461164959	5	4

4.2-12; CFC Client and Online Database

Using Android Studio, we built the CFC phone component as an exported jar file. We used an obfuscation tool, "**Proguard**" [54] that obfuscates our jar file targeting for example classes and variables names. We inspected the effect of this obfuscation using a decompiler tool called "**Java Decompiler**" [55]. 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 (figure 4.2-12) that hosts ads using this library, in order to generate a revenue on every ad click.

When the application starts, our library requests an ad to show from the ad network by sending the publisher's ID (Figure 4.2-11 – **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 our CFC services. Using the LAMP server [58], 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 to display on the publisher's application in JSON format (Figure 4.2-11 – **step 2**). After fetching the ad, the library parses the JSON response and displays the ad in a simple banner.

In currently existing ad fetching systems, ad networks follow an ad selection process where they decide which ad to send for display based on many factors such as the bidding placed on the ad by the advertiser, application's 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 thesis. 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.

When the user clicks on an ad, the library performs several sequential requests (Figure 4.2-11):

Step 3-4: For billing purposes, using the following API *request_billing_from_adnetwork(publisher_id, ad_id)*, the library informs the ad network that an ad is clicked, by sending the publisher's ID and the ad ID (Figure 4.2-11- step 3). After confirming this publisher-ad ID combination, the ad network returns a confirmation billing JSON response to the publisher (Figure 4.2-11 - step 4).

Steps 1, 2, 3 and 4 are performed in a very similar way to existing ad fetching systems. However, the following additional steps are presented in our system:

Steps 5-6-7 (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 (we assume 5 seconds) before exiting the advertised website. Therefore, the CFC client library sends a request to the CFC server to indicate that an ad view session has begun on the client side. This API defined as *request_ad_clicked_challenge(publisher_id, IP_address, state, challenge_token, session_id, timestamp)* 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 (the fields challenge_token and session_id are NULL in step 5). This non-local IP address is fetched on the client side using an external API called "*ipify*" [52]. The CFC server saves this information in its online database with a state field of value 1 and a timestamp. The server saves the ad-opened-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 using the same API with a state equals 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 hasn't exited the advertised website yet.

Step 8 (Ad View): After receiving a confirmation of the ad click challenge from the server, an Android native component known as a Webview is called and managed by the phone CFC component. The benefit of using Webview in this proposed system, is the ability to detect when the user exists the advertiser's website, by either clicking the native Android back button, or by clicking the Exit button. This Web view requests the corresponding ad URL and loads the advertised website in its view. We created a sample advertiser's website to test this step.

Step 9, 10, 11 (Ad closed challenge): Once the Webview is closed (by detecting the call of the *onbackpressed()* Android method), or the application is no longer visible (by detecting when it's 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 verify 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 the difference between the previous saved timestamp and the current received timestamp, which represents the duration spent on the advertiser's website. The CFC admin can determine the minimum duration that is required to consider this ad session as non-malicious, for example 5 seconds. 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 flags 4 and 5 are added to the CFCA for analysis (explained in section 4.2.).

4.2.1. 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 because they simply 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 detailed view, where it is able to track the user's engagement in each advertising website.

4.2.1.1.CFCA Dataset

To evaluate the performance of our algorithm, we generated the following dataset using Matlab:

We created 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 (not necessarily less than five seconds), and closed the ad webview after performing the handshake, i.e. ad request that completed the 11 CFC steps with state = 4 (duration > 5 seconds) or state = 5 (duration < 5 seconds). 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 he performs any type of click fraud by sequentially clicking on ads and exiting the advertiser's website (closing immediately the ad Webview), to open a new ad Webview and 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 five seconds. 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

TruthST (**RMC >= TruthST**). This TruthST is the lower limit used to classify the 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 truth and is different than our CFCA algorithm. 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 total number of ad requests per honest publisher is **TNHP**.

4.2.1.2. CFCA Classification

To simulate ads traffic in the real world, we generated 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 the real world, each iteration correspond to whenever the CFC administrator fetches ad requests rows 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: **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).

4.2.2. 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 spoofing IP) creates a sequential click automated tool that doesn't spend a long duration on the advertiser's website, since it's forced to exit the ad Webview to open another new ad immediately. *Defense*: 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 spoofing IP) places ads next to buttons, in order to trick user into clicking them. This fraudulent act is known as placement ads in the literature. *Defense*: Since such user is not necessarily interested in the ad, he 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 5 seconds on the advertised website. We consider these redirected ad requests as non-malicious since they did actually spend more than 5 seconds on the website and thus could become customers from the advertiser's point of view.

Type C: A malicious publisher (without spoofing IP), after completing the ad-clicked-challenge closes the Webview in less than 5 seconds, to be able to open a new ad session. However, being on the client's side, he is able to drop the request generated by the library on Webview closing (when duration spent was less than 5 seconds), and fabricate this request after 5 seconds has passed. The goal of this attack is to be able to sequentially click on ads to generate higher revenues without spending the required duration on the advertiser's website and without being detected. *Defense*: Although the attacker is able to drop the legitimate ad closing request and fabricate it, and by that to falsify the duration spent on the advertiser's website, however, since the IP is not spoofed, the CFC server can identify the user by IP 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 5 seconds on 3 different websites in a given time interval less than 15 seconds.

Type D: A malicious publisher, uses a spoofed IP address to be considered as a new user with each ad request. *Defense*: Whether this publisher fakes IP before or after completing the first ad-clicked-challenge (steps 5, 6, 7), since he did not complete both challenges, his state in the online database will not be updated to 4 (4 being the final state of the ad requests considered as non-malicious). As explained in section 4.2.2.1, the CFCA can simply time-out the ad requests with a state different than 4 and a reasonably old timestamp (to take into consideration honest sessions that still haven't 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. *Defense*: In addition to being forced to spend more than 5 seconds on each advertising website, the proposed system is able to identify the human clicker by his IP, which limits the number of fraudulent clicks per human clicker before being detected as malicious by our system.

4.3. Implementation

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 **5,000 < TNMP < 6,000**, and a total number of requests per honest publishers **1 < NHP < 1,500,** 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 [56], 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.

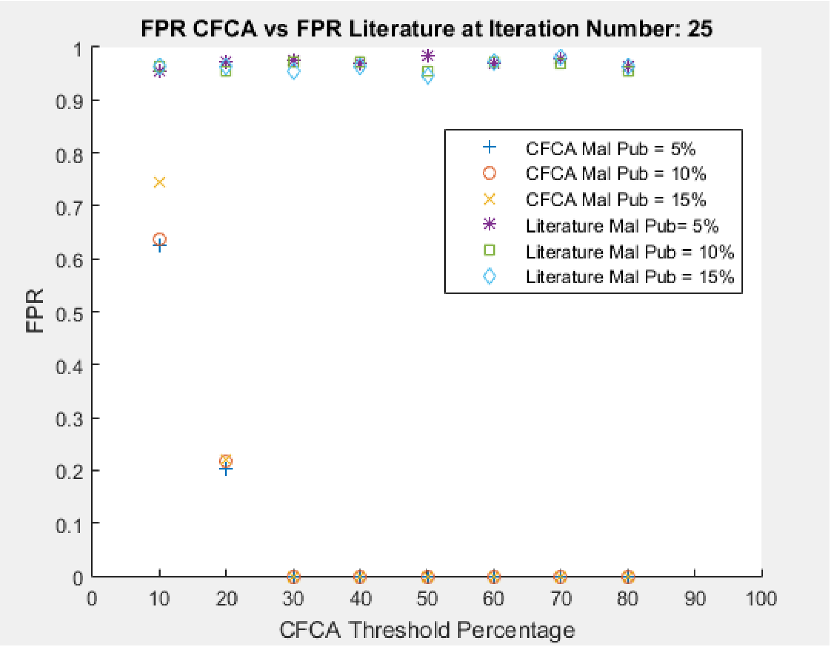
We evaluated our model under different scenarios: we tested three different percentages of malicious publishers **MPP = 5**, **MPP = 10** and **MPP = 15**. For each of these scenarios, we tested our CFCA using different CFCA starting point **CFCAST = [10,20,30,40,50,60,70,80].** We calculated the False Positive Rate FPR (figure 4.4-12), True Positive Rate TPR (figure 4.4-13), and Accuracy ACC (figure 4.4-14) in each iteration.

As part of their proposed methodology, the authors of [36] 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 seconds 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 seconds 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.

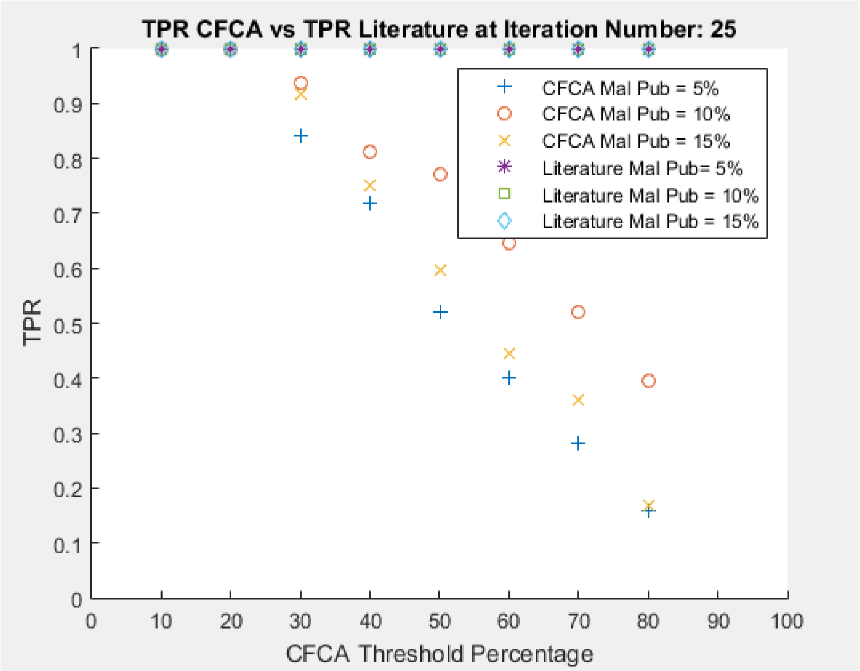
We compared our proposed method with their approach by using our CFCA algorithm with our base set, different CFCA starting point **CFCAST = [10,20,30,40,50,60,70,80]**, and a **starting point of 1** for their results (to simulate their per-click approach). Similarly to our method, we calculated for the literature, the False Positive Rate FPR (figure 4.4-12), True Positive Rate TPR (figure 4.4-13), and Accuracy ACC (figure 4.4-14) in each iteration.

4.4. Results and Analysis

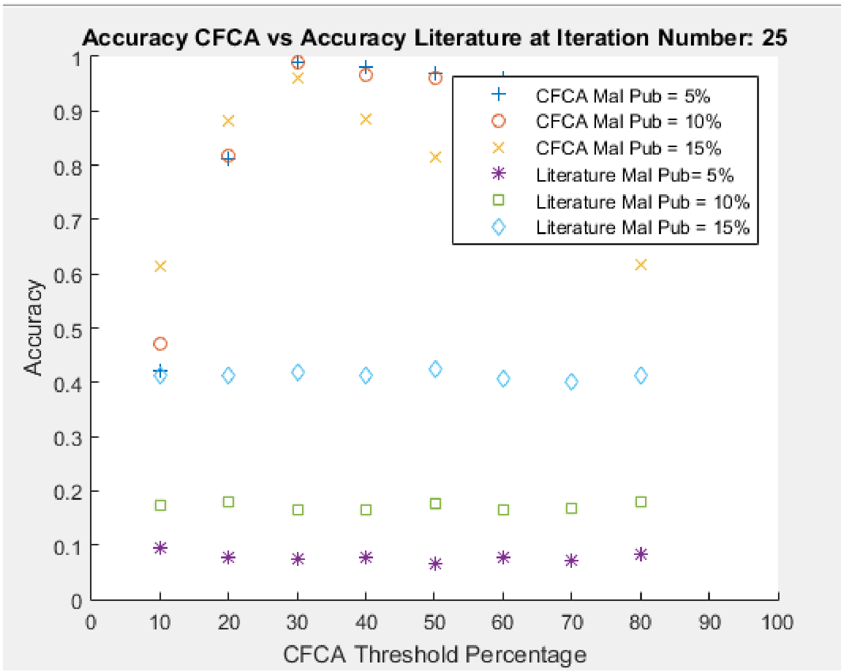
The results presented in figure 4.4-12, figure 4.4-13 and figure 4.4-14, correspond to the iteration number I = 25.



4.4-13: FPR CFCA vs FPR Literature at Iteration I = 25



4.4-14: TPR CFCA vs TPR Literature at Iteration I = 25



4.4-15: Accuracy CFCA vs Accuracy Literature at Iteration I = 25

Based on the figure 4.4-12, for a percentage of malicious publishers **MPP = 10**, our model presents a low false positive rate starting from **FPR = 0.21** when using

CFCA starting point of **CFCAST = 20**, and it decreases to **FPR = 0** with the increase of CFCA starting point to **CFCAST = [30, 40, 50, 60, 70, 80]**. As expected, the FPR calculated based on the literature approach 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 figure 4.4-13, 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 literature review remains **TPR = 1** for all the different scenarios.

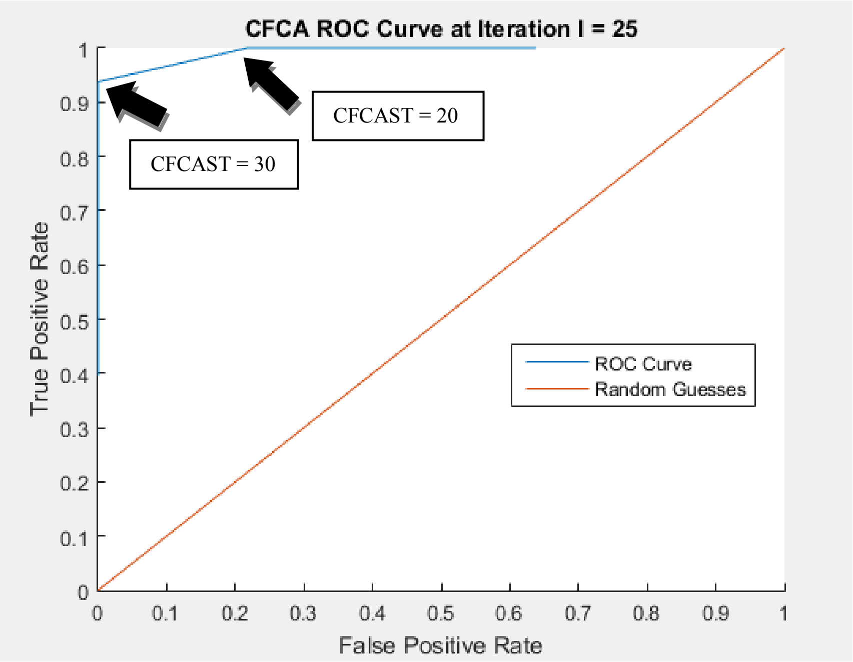
Based on the figure 4.4-14, our model presents a high accuracy **ACC = 0.82** when using a CFCA starting point of **CFCAST = 20**, and it increases to **ACC = 0.98** with the increase of the CFCA starting point to **CFCAST = 30**. It maintains this high level of accuracy for the rest of the CFCA starting points.

4.4.1.1. CFCA ROC Curve

Based on our results, we can conclude that changing the malicious publisher percentage **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** (figure 4.4-15). Depending on how aggressive we would like our model to be we can select between two CFCA starting points **CFCAST**

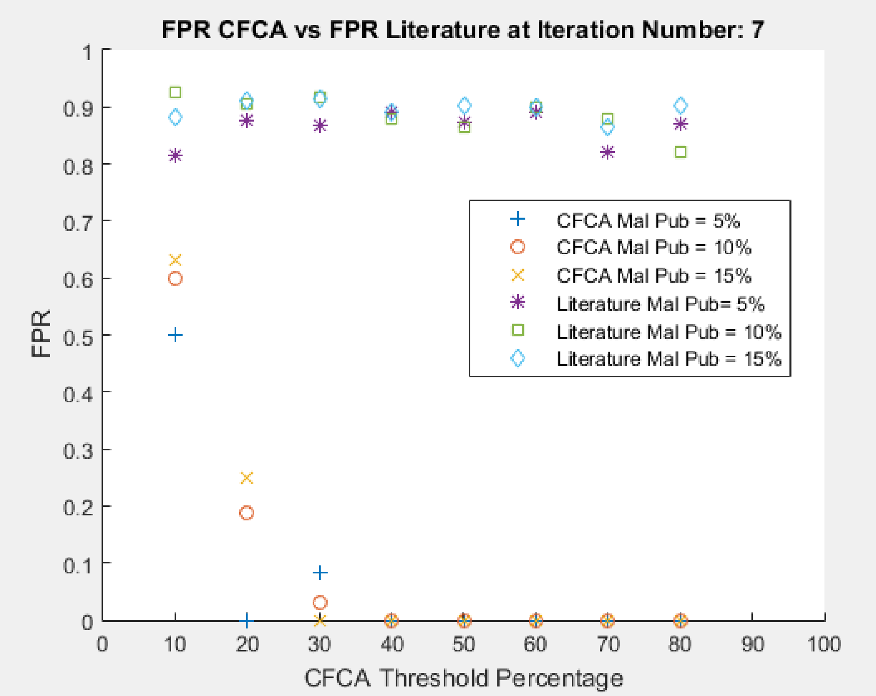
= 20 (TPR = 1, FPR = 0.21) or **CFCAST = 30 (TPR = 0.93, FPR = 0)**. Another method to determine which starting point to use is the F-measure (also known as F1 score) that takes into consideration the precision (also known as positive predictive value) and the recall (also known as the sensitivity or TPR). The F1 score when using **CFCAST = 20** is **F1Score = 0.6532** , whereas it's **F1Score = 0.967** when using **CFCAST = 30**.



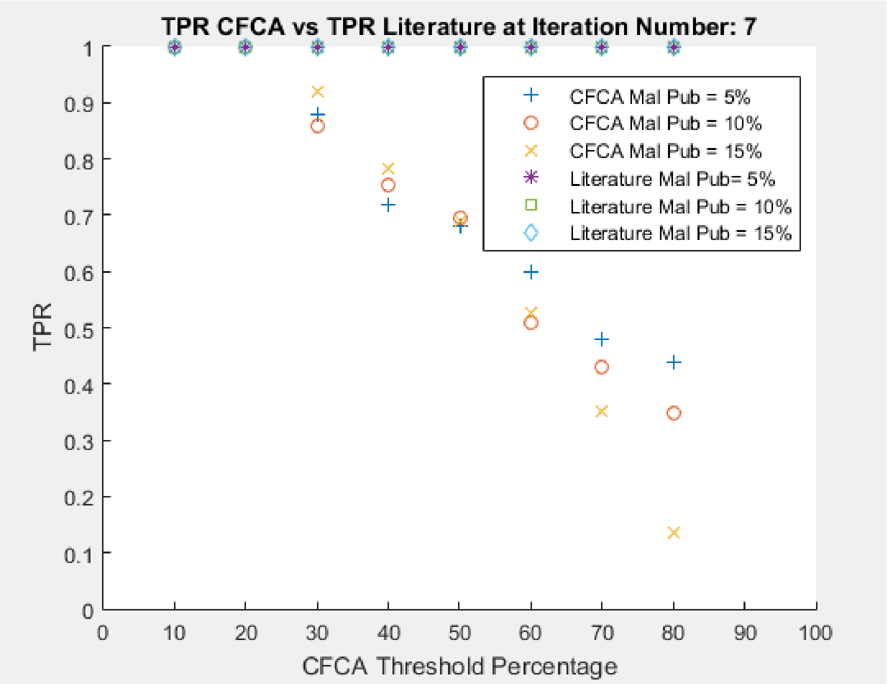
4.4-16: CFCA ROC Curve

4.4.1.2. CFCA Convergence

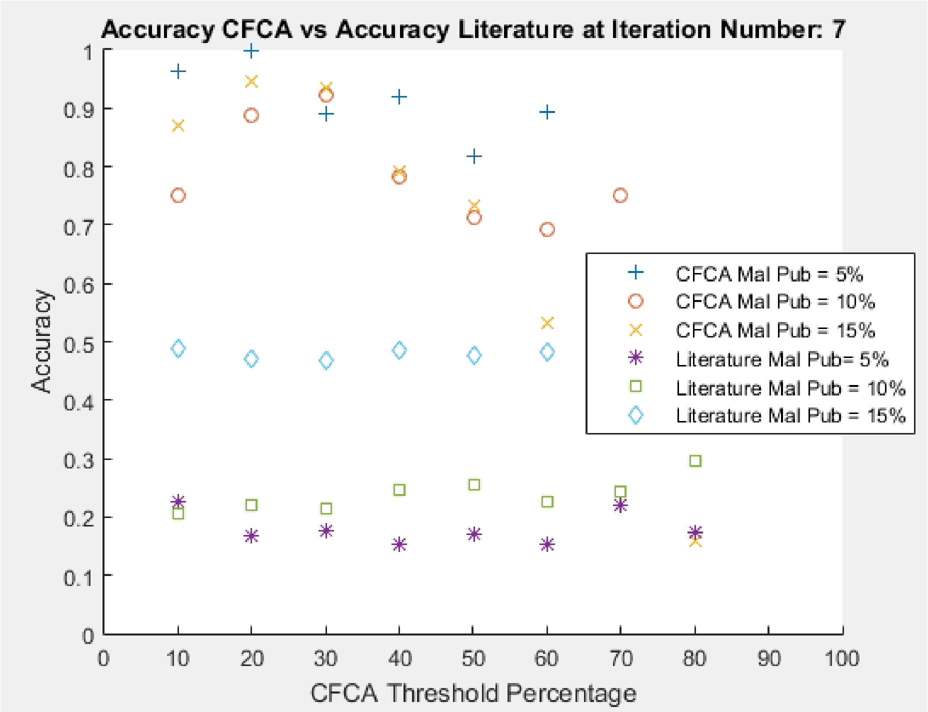
To evaluate how fast our proposed model is able to achieve a high true positive rate while maintaining a low false positive rate and a high accuracy level, we tested it at different low iterations (I = 5, I = 6, I = 7, I = 8, I = 10…). We conclude that our system converges starting at iteration **I = 7** (figure 4.4-16, figure 4.4-17, figure 4.4-18), whereas it presents Null FPR values at earlier iterations (I < 7). It is expected not 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.



4.4-17: FPR CFCA vs FPR Literature at Iteration: 7



4.4-18: TPR CFCA vs TPR Literature at Iteration: 7



4.4-19: ACC CFCA vs ACC Literature at Iteration: 7

4.4.1.3. CFC Duration Accuracy

To measure the accuracy of the duration captured by the CFC phone component, we clicked on the ad webview in our Android application (figure 4.2-12), closed the ad webview 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 phone component). We repeated this experiment **30** times for a duration **less than 5 seconds,** and **30** times for a duration **higher than 5 seconds** and calculated the accuracy of the duration in each of these experiment. As shown in table 4.4.1-8, the accuracy of the duration measured by our CFC phone component in both types of experiments is very high (**85.46%** and **90,73%**).

4.4.1-8: CFC Duration Accuracy

Duration	Average Accuracy(%)	Average (Sec)	STDEV(Sec)
Less than 5 seconds (30 experiments)	85.461	0.497	0.0896
More than 5 seconds (30 experiments)	90.737	0.799	0.0653

4.4.1.4.CFCA Resource Consumption

To measure the resource consumption of our CFC phone component, we created a sample Android application that displays a white background and our CFC banner ad. At run time, the CFC phone component fetches an ad and displays it in the CFC banner ad. We clicked on the ad, spent 5 seconds 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 opened/closed challenges or any other component implemented in our system that are not adopted by traditional ad networks such as the opening/closing of the Webview. Therefore, we used an experiment duration of 30 seconds that is sufficient enough 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 final result, we repeated the experiment 10 times (30 seconds duration each).

We created another Android application that integrates AdMob [15] 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 [15] opens either 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. 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 final result, we repeated the experiment 10 times (30 seconds duration each).

As shown in table 4.4.1-9, our proposed model consumes in terms of battery consumption on averag**e 0.7851 J/30 seconds**, whereas Admob [15] consumes on average **12.13 J/30 seconds**, and **3.03 J/ 30 seconds** to display the advertiser's application in the Google Play Store.

As shown in table 4.4.1-9, our proposed model consumes in terms of bandwidth consumption on averag**e 18.494 KB/30 seconds**, whereas AdMob [15] consumes on average **385 KB/30 seconds**, and **144.44 KB/ 30 seconds** 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 [15].

4.4.1-9: Resource Consumption Comparison

Ad Network	Bandwidth (KB/30 seconds)		Battery (J/30 seconds)	
	Average	STD	Average	STD
CFC	18.494	5.098	0.7851	0.213
AdMob [15]	385	26.13	12.13	2.24
Google Play Store	144.44	154.68	3.03	1.348

4.5. CFC Conclusion and future work

In this thesis, 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, 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 suitable 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 at the same time (TPR = 0.93 with CFCAST = 30). Our system is transparent to both the publisher and the advertiser in terms of implementation.

However, our framework represents 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 he is redirected after clicking the ad, to an application download page instead of an advertising website.

4.6. Summary

This chapter presents the second part of this thesis: Using crowdsourcing for click fraud detection. It explains the proposed solution, the implementation followed in both client and server side, the attackers’ models that are covered by CFC, and the proposed CPA model. In addition, it presents the obtained results of our experiments, and its analysis. In addition, it provides an insight on the future work that could enhance our CFC model and a conclusion of this section.

CHAPTER 5

CONCLUSION & FUTURE WORK

In the first part of the thesis we evaluated the bandwidth and battery consumption of in-app ads among several ad networks. The experimental procedure followed in this study demonstrates that resource consumption varies among networks and is usually consistent in the same ad network for a given duration. In addition, this study highlights a common behavior of fetching ads between several ad networks where ads are fetched at the beginning of runtime and displayed throughout the application session. This variation and significance of resource consumption indicates that developers need to take into consideration the bandwidth and battery consumptions when choosing the right ad network. On the other hand, the ad networks should adopt an ad-fetching mechanism that does not include large network overhead or unnecessarily heavy battery consumption. Furthermore, ad networks should adopt a transparent approach where they specify clearly the bandwidth and battery consumption associated with their ads fetching platform. Future work includes the evaluation of additional ad networks, testing on several devices, and a more accurate way to measure the variation of this consumption across time.

In the second part of the thesis, 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, 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 suitable 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 at the same time (TPR = 0.93 with CFCAST = 30). Our system is transparent to both the publisher and the advertiser in terms of implementation.

REFERENCES

[1] AppBrain, “Distribution of free vs. paid Android apps”, November 2014 [Online]. Available: http://www.appbrain.com/stats/free-and-paid-android-applications [Accessed: April 2016]

[2] Ash, cloud-iq, “In-App Advertising Growing in Popularity in UK”, May 2014 [Online].Available: http://blog.cloud-iq.com/blog/in-app-advertising-growing-in-popularity-in-uk [Accessed: April 2016]

[3] L. Zhang, D. Gupta, and P. Mohapatra, "How Expensive are Free Smartphone Apps?", in Mobile Computing and Communications Review, Vol. 16, No. 3, July 2012.

[4] A. Pathak, Y. C. Hu, and M. Zhang, "Where is the energy spent inside my app?: Fine grained energy accounting on smartphones with eprof," in Seventh ACM European Conference on Computer Systems (ACM EUROSYS), Bern, Switzerland, 2012.Fine grained energy accounting on smartphones with eprof," in Seventh ACM.

[5] A. Pathak, Y. C. Hu, M. Zhang, P. Bahl, Y-M Wang, "Fine-Grained Power Modeling for Smartphones Using System Call Tracing", in EuroSys’11, Salzburg, Austria, 2011.

[6] I. Prochkova, V. Singh, and J. K. Nurminen, "Energy Cost of Advertisements in Mobile Games on the Android Platform", in Sixth International Conference on Next Generation Mobile Applications, Services and Technologies, 2012.

[7] N. Vallina-Rodriguez, J. Shah, A. Finamore, Y. Grunenberger, H. Haddadi, K. Papagiannaki, and J. Crowcroft, "Breaking for Commercials: Characterizing Mobile Advertising", in IMC’12, Boston, Massachusetts, USA, 2012.

[8] X. Chen, A. Jindal, and Y. C. Farmington, "How Much Energy Can We Save from Prefetching Ads? Energy Drain Analysis of Top 100 Apps", in HotPower'13, Farmington, PA, USA, 2013.

[9] P.Mohan, S.Nath, O.Riva, "Prefetching mobile ads: Can advertising systems afford it?", in Eurosys'13, Czech Republic, 2013.

[10] A. J. Khan, K. Jayarajah, D. Han, A. Misra, R. Balan, and S. Seshan, "CAMEO: A Middleware for Mobile Advertisement Delivery", in MobiSys'13, Taipei, Taiwan, 2013.

[11] A. J. Khan, V. S., A. M., and S. Seshan, "Mitigating the True Cost of Advertisement- Supported "Free" Mobile Applications", in HotMobile, San Diego, 2012.

[12] S. Seneviratne, A. Seneviratne, and P. Mohapatra, "Personal Cloudlets for Privacy and Resource Efficiency in Mobile In-app Advertising", in MobileCloud'13, Bangalore, India. 2013.

[13] "Extensive List of Mobile Ad Network Companies", [Online]. Available: http://gulyani.com/complete-list-of-mobile-ad-networks-companies/ [Accessed: April 2016]

[14] "Mobile monetization models – making money out of apps", [Online]. Available: http://inspiredm.com/mobile-monetization-models-making-money-out-of-apps/ [Accessed: April 2016]

[15] AdMob, "Monetize, Promote and Analyze your apps with AdMob" [Online]. Available: http://www.google.com/ads/admob/ [Accessed: April 2016]

[16] Flurry, "Personalizing the mobile experience" [Online]. Available: http://www.flurry.com/about-flurry [Accessed: April 2016]

[17] Appia, “The leading mobile user acquisition network” [Online]. Available: http://www.appia.com/developers/ [Accessed: April 2016]

[18] AppFlood, “The future of mobile is programmatic” [Online]. Available: http://appflood.com/?url=http%3A//appflood.com/user/dashboard [Accessed: April 2016]

[19] RevMob, “Dashboard” [Online]. Available: https://www.revmobmobileadnetwork.com/dashboard [Accessed: April 2016]

[20] MobPartner, "Mobile Performance Marketing for Customer Acquisition and Engagement" [Online]. Available: http://www.mobpartner.com/ [Accessed: April 2016]

[21] Millennial Media, "Millennial Media’s Mobile Audience Solutions connects brands with their target consumers at the moments that matter most" [Online]. Available:http://www.millennialmedia.com/ [Accessed: April 2016]

[22] InMobi, "The Only Mobile-First Advertising Platform with 1 billion mobile uniques" [Online]. Available:http://www.inmobi.com/ [Accessed: April 2016]

[23] MoPub, "Drive More Mobile Ad Revenue" [Online]. Available:http://www.mopub.com/ [Accessed: April 2016]

[24] MobFox, "Introducing: MobFox Native Ads" [Online]. Available:http://www.mobfox.com/ [Accessed: April 2016]

[25] venturebeat, "Interstitial mobile ads are killing it: 25 times video views, 7 times conversions, 9 times revenue" [Online]. Available:http://venturebeat.com/2013/08/09/interstitial-mobile-ads-are-killing-it-25x-video-views-7x-conversions-9x-revenue/ [Accessed: April 2016]

[26] LeadBolt, "The Next Generation Mobile Marketing Platform" [Online]. Available:http://www.leadbolt.com/ [Accessed: April 2016]

[27] Airpush, " Airpush:" [Online].Available:http://www.airpush.com/ [Accessed: April 2016]

[28] Admoda, "Admoda", [Online].Available: http://www.admoda.com/ [Accessed: April 2016]

[29] Smaato, "About Smaato", [Online].Available: https://www.smaato.com/company/ [Accessed: April 2016]

[30] GreyStripe, "Android SDK Integration", [Online].Available: http://support.greystripe.com/android [Accessed: April 2016]

[31] madvertise, "Premium mobile advertising", [Online].Available: http://madvertise.com/ [Accessed: April 2016]

[32] J. Gui, S. Mcilroy, M. Nagappan, and W. G.J. Halfond. "Truth in Advertising: The Hidden Cost of Mobile Ads for Software Developers", In Proceedings of the 37th International Conference on Software Engineering (ICSE), May 2015.

[33] H. M. Dermanilian, F. Saab, I. H. Elhajj, A. Kayssi, A. Chehab, "Energy-Efficient Security for Voice over IP", in International Journal of Network Security, Vol.17, No.1, PP.11-26, Jan. 2015

[34] AT&T, "Create your Mobile Share Value plan", [Online].Available: http://www.att.com/shop/wireless/data-plans.html#fbid=EhuxYcdIz02 [Accessed: April 2016]

[35] dmnews, "Bots Mobilize", [Online].Available: http://www.dmnews.com/mobile-marketing/bots-mobilize/article/291566/ [Accessed: April 2016]

[36] V. Dave, S. Guha and Y. Zhang, 'Measuring and fingerprinting click-spam in ad networks', in ACM SIGCOMM Computer Communication Review, vol. 42, no. 4, p. 175, 2012, Helsinki, Finland.

[37] “The Alleged $7.5 Billion Fraud in Online Advertising”, Online. Available: https://moz.com/blog/online-advertising-fraud [Accessed: April 2016]

[38] G. Cho, J. Cho, Y. Song, H. Kim (2015, August). ‘An empirical study of click fraud in mobile advertising networks’, in Availability, Reliability and Security (ARES), 2015 10th International Conference on (pp. 382-388), Toulouse, France, IEEE.

[39] J. Crussell, R. Stevens, H. Chen (2014, June). ‘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.

[40] T. Blizard,, N. Livic, (2012, October). ‘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.

[41] S. Alrwais, S. A., A. Gerber, C.W. Dunn, O. Spatscheck, M. Gupta, E. Osterweil (2012, December). ‘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.

[42] B. Stone-Gross, R. Stevens, A. Zarras, R. Kemmerer, C. Kruegel, G. Vigna (2011, November). ‘Understanding fraudulent activities in online ad exchanges’, in Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference (pp. 279-294), Berlin, Germany, ACM.

[43] P. Pearce, V. Dave, C. Grier, K. Levchenko, S. Guha, D. McCoy, G.M. Voelker (2014, November). ‘Characterizing large-scale click fraud in zeroaccess’, in Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (pp. 141-152), Arizona, USA, ACM.

[44] B. Miller, P. Pearce, C.Grier, C. Kreibich, V. Paxson (2011). ‘What’s clicking what? techniques and innovations of today’s clickbots’, in Detection of Intrusions and Malware, and Vulnerability Assessment (pp. 164-183), Amsterdam, The Netherlands, Springer Berlin Heidelberg.

[45] B. Liu, S. Nath, R. Govindan, J. Liu (2014, April). ‘DECAF: detecting and characterizing ad fraud in mobile apps’, in Proc. of NSDI, Seattle, WA, USA.

[46] A. Juels, S. Stamm, M. Jakobsson (2007, August). ‘Combating Click Fraud via Premium Clicks’, in USENIX Security (pp. 1-10), Santa Clara, CA, USA.

[47] D. Vasumati, M.S.Vani, R. Bhramaramba, O.Y. Babu. (April 2015). ‘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).

[48] W.Li, H. Li, H. Chen, Y. Xia (2015, May). ‘AdAttester: Secure Online Mobile Advertisement Attestation Using TrustZone’, in Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services (pp. 75-88), Florence, Italy, ACM.

[49] H. Haddadi (2010). ‘Fighting online click-fraud using bluff ads’, in ACM SIGCOMM Computer Communication Review, 40(2), 21-25, New Delhi, India.

[50] H. Xu, D.Liu, A. Koehl, H. Wang, A. Stavrou (2014). ‘Click Fraud Detection on the Advertiser Side’, in Computer Security-ESORICS 2014 (pp. 419-438), Wroclaw, Poland, Springer International Publishing.

[51] M. Vani, R. Bhramaramba, D. Vasumati, O.Y. Babu. ‘A Node-Tag Prediction Algorithm for Touch Spam Detection in Mobile Ad Networks’, in International Journal of Computer Engineering and Applications, Volume VIII, Issue III, Part I, December 2014.

[52] “ipify”, Available: https://api.ipify.org/, [Online].Available: [Accessed: April 2016]

[53] LAMP Server, “https://help.ubuntu.com/community/ApacheMySQLPHP” [Online].Available: [Accessed: April 2016]

[54] Proguard, [Online].Available: http://developer.android.com/tools/help/proguard.html [Accessed: April 2016]

[55] Java Decompiler, [Online].Available: “http://jd.benow.ca/”, [Accessed: April 2016]

[56] Business Insider, [Online].Available: “http://www.businessinsider.com/its-more-likely-you-will-survive-a-plane-crash-or-win-the-lottery-than-click-a-banner-ad-2011-6”, [Accessed: April 2016]

[57] Pechuán, L. M., Ballester, E. M., & Carrasco, J. M. G. “Online Advertising and the CPA Model: Challenges and Opportunities”, in International Journal of Engineering and Management Research, Volume-4, Issue-3, June-2014.

[58] Mahdian, Mohammad, and Kerem Tomak. "Pay-per-action model for on-line advertising." *International Journal of Electronic Commerce* 13.2 (2008): 113-128.

[59] 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.

[60] Hu, Yu, Jiwoong Shin, and Zhulei Tang. "Incentive Problems in Performance-Based Online Advertising Pricing: Cost per Click vs. Cost per Action." *Management Science* (2015).

[61] Ross, Kevin, and Kristin Fridgeirsdottir. "Cost-Per-Impression and Cost-Per-Action Pricing in Display Advertising with Risk Preferences." MANUFACTURING & SERVICE OPERATIONS MANAGEMENT (2011)

[62] Dellarocas, Chrysanthos. "Double marginalization in performance-based advertising: Implications and solutions." *Management Science* 58.6 (2012): 1178-1195.

[63] Hu, Y. J., Shin, J., & Tang, Z. (2010, January). 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.

[64] Agarwal, N., Athey, S., & Yang, D. (2009). Skewed bidding in pay-per-action auctions for online advertising. The American Economic Review, 441-447.

[65] Nazerzadeh, H., Saberi, A., & Vohra, R. (2008, April). Dynamic cost-per-action mechanisms and applications to online advertising. In Proceedings of the 17th international conference on World Wide Web (pp. 179-188). New York, ACM.

[66] Ding, Xuhua. "A hybrid method to detect deflation fraud in cost-per-action online advertising." Applied Cryptography and Network Security. Springer Berlin Heidelberg, 2010.

[67] "Ad Networks vs Ad Exchanges", [Online].Available: http://www.adbalance.com/ad-networks-vs-ad-exchanges/ [Accessed: April 2016]

[68] "Fraud from bots represents a loss of $6 billion in digital advertising" , [Online].Available: http://www.reuters.com/article/us-advertising-fraud-study-idUSKBN0JN0AW20141209 [Accessed: April 2016]