Privacy-preserving targeted mobile advertising:A Blockchain-based framework for mobile ads

SUBMISSION DATE / POSTED DATE

22-08-2020 / 02-09-2020

CITATION

Ullah, Imdad (2020): Privacy-preserving targeted mobile advertising:A Blockchain-based framework for mobile ads. TechRxiv. Preprint. https://doi.org/10.36227/techrxiv.12845717.v1

DOI

10.36227/techrxiv.12845717.v1

Privacy-preserving targeted mobile advertising: A Blockchain-based framework for mobile ads

Imdad Ullah, *Member, IEEE,* Salil S. Kanhere, *Senior Member, IEEE,* and Roksana Boreli

Abstract—The targeted advertising is based on preference profiles inferred via relationships among individuals, their monitored responses to previous advertising and temporal activity over the Internet, which has raised critical privacy concerns. In this paper, we present a novel proposal for a Blockchain-based advertising platform that provides: a system for privacy preserving user profiling, privately requesting ads from the advertising system, the billing mechanisms for presented and clicked ads, the advertising system that uploads ads to the cloud according to profiling interests, various types of transactions to enable advertising operations in Blockchain-based network, and the method that allows a cloud system to privately compute the access policies for various resources (such as ads, mobile user profiles). Our main goal is to design a decentralized framework for targeted ads, which enables private delivery of ads to users whose behavioral profiles accurately match the presented ads, defined by the ad system. We implement a POC of our proposed framework i.e. a *Bespoke Miner* and experimentally evaluate various components of Blockchain-based *in-app* advertising system, implementing various critical components; such as, evaluating user profiles, implementing access policies, encryption and decryption of users' profiles. We observe that the processing delay for traversing policies of various tree sizes, the encryption/decryption time of user profiling with various key-sizes and user profiles of various interests evaluates to an acceptable amount of processing time as that of the currently implemented ad systems.

Index Terms—Privacy, Advertising, Blockchain, Mobile Advertising System, Access Policy, Cryptocurrency

1 Introduction

Mobile advertising ecosystem is one of the most successful markets, in recent years, with billions of mobile devices, including smartphones, tablets, and computer tablets, and millions of mobile applications (apps) registered in various app platforms, such as Google Play Store, Apps Store etc. The advertising companies enable user tracking within apps, and hence profiling users, where user's personal information plays an important role in ads targeting. However, significant privacy concerns are associated with it, as suggested in a number of studies [1], [2], [3], [4], [5], [6]. Other studies [7], [8] indicate, unless consumers have specifically consented to it, that they have a direct relationship between consumer attitude and their behavior, i.e. consumers' trust in privacy of mobile advertising is positively related to their willingness to accept mobile advertising. The American ad industry, implemented self-regulated by implementing AdChoices[1] program, which states that consumer could *opt-out* of *targeted* advertising via online choices to control ads from other networks. However, another study [9] examines that *opt-out* users cause 52% less revenue (hence presents less relevant ads and lower click through rates) than those users who allow *targeted* advertising. In addition, the authors noted that these ad impressions were only requested by 0.23% of American consumers.

Hence, the purpose of *targeted* advertising is to deliver most relevant ads to achieve better view/click through rates and without exposing users' private information to the advertising companies and third party publishers/ad networks. Prior research works show the extent to which consumers are effectively tracked by a number of third parties and across multiple apps [10], mobile devices leaking Personally Identifiable Information (PII) [11], [12], apps accessing user's private/sensitive information through well defined APIs [13], inference attacks based on monitoring ads [6] and other data platform such as eXelate[2], BlueKai[3], and AddThis[4] that collect, enrich and resell cookies.

There are other works on protecting user's privacy by obfuscating user profiles[5] [5], [15] and based on security techniques [2], enabling user privacy [16], [17] by suggesting design changes to Android location API based on laboratory study [18] to understand developer's behavior. Other privacy preserving advertising systems use a mix of cryptography techniques and obfuscation mechanisms, e.g., [19], [20] and *targeted* mobile coupon delivery scheme using Blockchain [21]. We note that the majority of the private advertising systems have been proposed for browser based ads [22], [23], [24], [25] whereas only few works address the *in-app targeted* ads [2], [15], [21], [26]. Our focus in this work is to protect individual's privacy by decoupling all direct

- *I. Ullah is with the College of Computer Engineering and Sciences, Prince Sattam bin Abdulaziz University, Al-Kharj 11942, Saudi Arabia. E-mail: i.ullah@psau.edu.sa*
- *S. S. Kanhere is with UNSW Sydney, Australia. E-mail: salil.kanhere@unsw.edu.au*
- *R. Boreli was with CSIRO Data61 Sydney, Australia. E-mail: roksana@tmppbiz.com*

1. Founded in 2010. https://optout.aboutads.info/?c=2&lang=EN

2. https://microsites.nielsen.com/daas-partners/partner/exelate/

3. https://www.oracle.com/corporate/acquisitions/bluekai/

4. https://www.addthis.com/

5. A large scale investigation of obfuscation use where authors analyze 1.7 million free Android apps from Google Play to detect various obfuscation techniques is presented in [14].

communications between apps, analytics companies, which they use for *targeted* advertisements, advertising systems (including third party) for their activities (both for web and apps activity) using Blockchain technology.

Blockchain is based on a distributed ledger of transactions, shared across the participating entities, and provides auditable transitions [27], where the transactions are verified by participating entities within operating network. Among the participating entities; the *Miner* is a special node responsible for generating transactions, adding them to the pool of pending transactions and organizing into a *block* once the size of transactions reaches a specific *block size*. The process of adding a new block to the Blockchain is referred to as *mining* and follows a consensus algorithm, such as Proof of Work (POW) [28] and Proof of Stake (POS) [29], which ensures the security of Blockchain against malicious (*Miner*) users. The participating entities use the *Public-Private Key* pair that is used to achieve the anonymity [30]. Among various salient features of Blockchain, i.e. irreversible, auditable, updated near real-time, chronological and timestamp, which, in addition, disregards the need of a central controlling authority; thus making it a perfect choice for restricting communication between the mobile apps and the analytics/ad companies and keeping individual's privacy.

Our main contributions in this paper include the following. We investigate the relation between the mobile apps usage and the resulting user profiles derived by a major analytics network i.e., Google AdMob. We mainly describe three states of the user profiles i.e. the user profiles that are established during when no interests are derived by the analytics services i.e. *profile establishment* process, the state of the profiles once apps are used that would originate interests other than the existing set of apps i.e. *profile evolution* process, and *stable states* of the profiles where subsequent use of existing set of apps have no effect over the profiles. We also determine the broad profiling rules during various processes of utilising android apps: that profiles represent an aggregation of interests generated by use of individual apps, that there is a fixed mapping of apps to sets of interests and that a minimum level of activity (in a period of up to 72h) is required to build a *stable* user profile. We correspondingly develop an analytical model for depicting various profiling states and rules to represent profiling processes.

Following, we design a framework for secure user profiling and Blockchain-based *targeted* advertising system for *in-app* mobile ads and the design of various critical components of advertising system, as follows: the **1**. Ads Placement Server (`APS`) for evaluating ads according to various profile interests, hashing ads and sending hashed profiles along with corresponding ads to `CS` **2**. Cloud System (`CS`) for storing hashed profiles along with corresponding ads sent by `APS`, in addition, maintaining a copy of user profiles **3**. Cluster Head (`CH`) for processing *ad-block* requests, *access control policy* for various operations, such as storing user profile etc., and fetching ads from `CS` according to user profiles **4**. System App (*Miner*) that sits on user's mobile device and calculates user's profiles, sends hashed user profile to `CH`, uploads user profiles on `CS`, and requests and delivers ads to mobile app **5**. Billing Server (`BS`) for calculating *billing* for presented and clicked ads with the use of Cryptocurrency for ensuring privacy, secure payment and for compatibility with the underlying proposed advertising system over Blockchain **6**. *Access Control Policy* implemented in *Miner* for holding an *access control list* that allows apps regarding evaluation of different operations e.g. to check whether an app is allowed to request for ads, participate in *billing* etc., and on `CS` to control access of *ad-request* coming from various *Miners*. The *access policy* is implemented as a *policy tree* that consists of matching functions for allowing access to various operations or resources and their corresponding actions.

We design the structure for various transactions for an *ad-request/response block* for operating within the Blockchain network e.g. block structure of an `Ad-Block-Header-Request`, `Ad-Block-Header-Response`, and `Ad-Block-Structure`. We design various algorithms that specify comprehensive operations of various components of the advertising system. Furthermore, we discuss various design requirements in terms of privacy, reduced overhead over (user) client, compatibility of *billing* with Cryptocurrency, and achieving system efficiency and component's compatibility within the Blockchain network.

We develop an Android-based Proof of Concept (POC) implementation i.e. a *Bespoke Miner* for evaluating various components of Blockchain-based *in-app* advertising system, such as, generating *public-private key* pair, evaluating user profiles, encrypting user profile or decrypting the ads received from `CS`, implementing *access policy*, and its interaction with the `CS`. We design a detailed experimental setup that consists of *Bespoke Miner* and `CS` (a local machine for correspondence with the *Bespoke Miner*), following, we utilise real parameter settings obtained by extensive our real-time experiments [15], [31] (and further verified that these parameters still hold) that include various sizes of an ad, set of interests with a user profile etc. We calculate the processing delay for generating *public-private key* pair, encryption/decryption time of user profiling with various key-sizes and user profiles, and processing delay for traversing *policies* of various sizes. We observe that, using various $512, 1024, 2048, 4096$ bits sizes of *public-private key* pairs, the Min processing time ranges from 0.005sec to 0.30sec.

Following, we calculate encryption and decryption time for user profiles with interests, randomly chosen up to 20 interests with an Avg and $St.Dev$ of respectively 10.53 and 5.88 interests, using $1024, 2048, 4096, 8192$ bits sizes of *public-private key* pairs and evaluate the processing delays of $Avg, Min, Max, St.Dev$. We observe that, for instance, the Min and Max profile encryption times with $(1024, 8192)$ bits key sizes are respectively (0.1msec, 1.2msec) and (2.2msec, 16.3msec), with an Avg and $St.Dev$ of (0.4msec) each and (6.0msec, 3.2msec). In addition, the decryption time evaluates to (0.001sec, 0.005mse) and (0.35sec, 1.67sec) with Avg and $St.Dev$ of (0.002sec, 0.0009sec) and (0.8sec, and 0.36sec). Based on our observations [2], [5], we note that this is an (equivalent) acceptable amount of processing time to that of when a mobile app requests ad in the beginning of an app launch or during the app usage session.

Rest of the paper is organised as follows: Section 2 presents detailed discussion over the profiling process and formulates the problem. Section 3 presents various

components of the proposed framework for secure user profiling and Blockchain-based *targeted* advertising system for *in-app* mobile ads. In Section 4, we present the `Ad-Block-Structure` for an *ad-request/response* for operating within the Blockchain network. Details of `Ad-Block` operations is presented in 5, followed by experimental setup, including details of POC implementation for bespoke *Miner*, and experimental evaluation in Section 6. Related work is summarised in Section 7 and conclusion of our work is given in Section 8.

2 Problem Formulation

We formalise the system model that consists of the applications' profiles, interests' profiles and the conversion of resulting profiles by use of applications in an app profile. In particular, we provide the insights of *establishment* of *Interests profiles* by individual apps in the *Apps profiles* and then show how the profiles *evolve* when some apps other than the initial set of apps are utilised.

2.1 Profiling

An app marketplace includes a set of $\mathcal{A}$ mobile apps, that are commonly (e.g., in Google Play or in the Apple App store) organised within different categories. We denote an app by $a_{i,j}$, $i = 1, ..., A_j$, where A_j is the number of apps that belong to an app category Φ_j, $j = 1, ..., \tau$ and τ is the number of different categories in the marketplace.

A user may be characterised by a combination of apps installed on their mobile device(s), comprising a subset S_a of the apps set $\mathcal{A}$. An *Apps profile* K_a can therefore be defined as:

$$K_a = \{\{a_{i,j}, \Phi_j\} : a_{i,j} \in S_a\} \tag{1}$$

Analytics companies (e.g., Google or Flurry) profile users by defining a set of profile interests $\mathcal{G}$, i.e., characteristics that may be assigned to users. Interests are commonly grouped into categories, with interests $g_{k,l}$, $k = 1, ..., G_l$, where G_l is the number of interests that belong to an interest category Ψ_l, $l = 1, ..., \epsilon$. ϵ is the number of interest categories defined by the analytics company. Table 1 shows notations used in this work.

Profiling characterises the user by the combination of their interests, i.e., by a subset S_g of the full interest set $\mathcal{G}$. Google profile interests[6] are grouped, hierarchically, under vaiours interests categories, with specific interests. An *Interests profile* I_g for a user can therefore be defined as:

$$I_g = \{\{g_{k,l}, \Psi_l\} : g_{k,l} \in S_g\} \tag{2}$$

We note that although various types of information may be used to generate user profiles, here we focus on the interests derived from the usage of installed apps. Note that we will use the *Apps profile* K_a and *Interests profile* I_g respectively for calculating the *Contextual* and *Targeted* ads.

In addition, *ads targeting* is based on demographics so as to reach a specific set of potential customers that are likely to be within a specific age range, gender etc., Google[7] presents a detailed set of various demographic *targeting* options for ads display, search campaigns etc. Hence, profiling also characterises users by various demographics under different options. The demographics D are usually grouped into different categories, with specific options $d_{m,n}$, $m = 1, ..., M$ and $n = 1, ..., N$, where M is the targeted demographics ranges, e.g. '18-24', '25-34', '35-44', '45-54', '55-64', '65 or more', and 'Male', 'Female', 'Rather not say', and N is the number of different demographic *targeting* options e.g. household income, parental status, location etc. Note that for simplicity, we do not consider the user history and leave it for future work. We suggest that the profiling is done on the user side (i.e. done by the *System App*[8], see Figure 2).

Following sub-section details derivation of *Interests profile*, which we call the profile *Establishment* process.

2.2 Mapping Rules for Profiling

We now deliberate the profiling rules as a result of utilisation of mobile apps. The advertising networks derive *Interest profiles* for wide range of audiences based on the collected information and accordingly *target* the mobile users. Note that the mapping rules outlined in this section are based on our observations [5] [31] [6].

2.2.0.1 **Profile Establishment**: User profile is established by using the installed apps for certain amount of time, after it has sufficient interactions with analytics services. We assume that there is a mapping of *Apps profile* to an *Interests profile* defined by Google Analytics, which is used by analytics companies to individually characterise user's interests across the advertising ecosystem. A sample mapping is also given in [5] where we observed mapping of *Apps profile* to *Interests profile* by a series of various *profiling* experiments. This mapping includes the conversion of apps categories Φ_j to interest categories Ψ_l. This mapping converts an app $a_{i,j} \in S_a$ to interest set $S_g^{i,j}$ after a specific level of activity t_{est}. The t_{est} is the *establishment threshold* i.e. time an app should be used in order to establish profile's interests. The set $S_g^{i,j}$ derived by an app $a_{i,j}$ can be represented as a set of interests $g_{k,l}$ along with their corresponding categories Ψ_l:

$$S_g^{i,j} \subset \{\{g_{k,l}, \Psi_l\} : g_{k,l} \in G\} \tag{3}$$

We note that all installed apps, from a specific mobile device, do not necessarily contribute to the profile[9], in which case the $S_g^{i,j}$ is an empty set.

Finally, the resulting *Interests profile* I_g is established as the combination of individual interests derived by each app. The *Interests profile* can be expressed as the union of all the interests sets $S_g^{i,j}$ derived by individual app $a_{i,j}$ in S_a; hence Eq. 2 can be re-written as follows:

$$I_g = \bigcup_{\forall (i,j) \,:\, a_{i,j} \in S_a} S_g^{i,j} \tag{4}$$

6. Google profile interests are listed in https://adssettings.google.com/authenticated?hl=en, displayed under the 'How your ads are personalized', other options and Google services can also be verified on Google Dashboard https://myaccount.google.com/dashboard?hl=en.

7. Demographic Targeting https://support.google.com/google-ads/answer/2580383?hl=en

8. The *System App* is similar to the AdMob SDK [5] that communicates with Google Analytics for deriving user profiles.

9. The full list of installed apps on an Android based phone is included in the library.db file /data/data/com.android.vending/databases/library.db

Symbols	Description
$\mathcal{A}$	Set of apps in a marketplace
τ	Number of app categories, Φ_j is a selected category, $j = 1, ..., \tau$
S_a	Subset of apps installed on a user's mobile device
$a_{i,j}$	An app $a_{i,j} \in A$, $i = 1, ..., A_j$, $j = 1, ..., \tau$, A_j is the number of apps in Φ_j
K_a	App profile consisting of apps $a_{i,j}$ and their categories Φ_j
$\mathcal{G}$	Set of interests in Google interests list
Ψ_l	Interest category in $\mathcal{G}$, $l = 1, ..., \epsilon$, ϵ is the number of interest categories defined by Google
S_g	Subset of Google interests in $\mathcal{G}$ derived by S_a
I_g	Interest profile consisting of $g_{k,l}$, $g_{k,l} \in S_g$
$g_{k,l}$	An interest in I_g, $k \in G_l$, $l \in \epsilon$
$S_g^{i,j}$	Set of interests derived by an app $a_{i,j}$
D	The set of demographics defined by analytics companies
$d_{m,n}$	Demographics options $d_{m,n} \in D$, $m = 1, ..., M$ and $n = 1, ..., N$
S_d	Subset of user's demographics derived by *System App*

TABLE 1: List of Notations.

Furthermore, *System App* derives users' demographics S_d from their settings (e.g. Gmail settings) or by inferring their demographics based on their activities. We suggest that the users may 'edit' their demographics within the *System App*, i.e. similar to that by visiting Google Ad Settings[10].

$$I_g = \bigcup_{\forall(i,j):a_{i,j}\in S_a} s_g^{i,j} \cup \bigcup_{\forall(m,n):m,n\in S_d} d_{m,n} \tag{5}$$

2.2.0.2 **Profile Evolution**: The profile is updated, and hence the *ads targeting*, each time variations in the users' behaviour are observed; such as for a mobile user using apps that would map to interests other than the existing set of interests. Let a user uses new set of apps S'_a, which has no overlap with the existing set of apps S_a that has created I_g i.e., $S'_a \subset \mathcal{A} \setminus S_a$.

The newly added set of apps S'_a is converted to interests $g_{k,l}$ with t_{evo} as *evolution threshold* i.e. the time required to evolve profile's interests. Thus, an app $a'_{i,j}$ in S'_a should be used for t_{evo} time in order to evolve profile's interests $S_g^{'i,j}$. The sets of interests $S_g^{'i,j}$ derived by $a'_{i,j}$ are represented as the union of all interests I'_g, i.e. $I'_g = \bigcup_{\forall(i,j):\, a_{i,j}\in S'_a} S_g^{'i,j}$ along with the updated demographics (if any) $\bigcup_{\forall(m,n):m,n\in S_d} d'_{m,n}$. Hence, the final *Interests profile*, I_g^f, after the profile *Evolution* process, is the combination of older interests derived during the profile *Establishment* I_g and during when the profile evolves I'_g and is given as:

$$I_g^f = I_g \cup I'_g \tag{6}$$

2.2.0.3 **Profile Development Process**: We now elaborate insights of profile *Establishment* and *Evolution* processes performed by Google analytics and adopt the same strategy of profile *Development* process. In order for the *Apps profile* to *establish* an *Interests profile*, a minimum level of activity of the installed apps is required. Furthermore, in order to generate one or more interests, an app needs to have the AdMob SDK. This was verified by testing a total of 1200 apps selected from a subset of 12 categories, for a duration of 8 days, among which 1143 apps resulted the *Interest profiles* on all test phones indicating "Unknown" interests. We also note that the *Apps profile* deterministically derives an *Interests profile* i.e., a specific app constantly derives identical set of interests after certain level of activity. We further note that the level of activity of installed apps be within a minimum of 24h period (using our extensive experimentations; we note that this much time is required by Google Analytics in order to determine ones' interests), with a minimum of, from our experimentations, $24/n$ hours of activity of n apps. For a sophisticated profiling, a user might want to install and use a good number of apps that would represent one's interests. After the 24h period, the profile becomes *Stable* and further activity of the same apps does not result in any further changes.

Similarly, during the profile *Evolution* process, the *Interests profile* starts changing by adding new interests; once apps other than the existing set of apps S_a are utilised. However, instead of 24h of period of evolving a profile, we observe that the *Evolution* process adds additional interests in the following 72 hours of period, after which the aggregated profile i.e. I_g^f becomes *Stable*. In order to verify the stability of the aggregated profile, we run these apps on 4th day; henceforth we observe no further changes. The mapping of *Apps profile* to *Interests profile* during the *Establishment* and during the *Evolution* process along with their corresponding *Stable* states are shown in Figure 1.

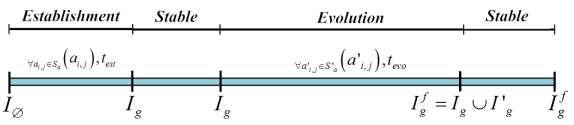


Fig. 1: Profile *Establishment* & *Evolution* processes. $I_\emptyset$ is the empty profile before apps utilisation. During *Stable* states, *Interest profiles* I_g or I_g^f remains the same and further activities of the same apps have no effect over the profiles.

2.3 Problem Statement

The problem addressed in this paper is where advertising systems profile and track mobile users for their activities and to tailor ads to their specific interests. In complex mobile environment, as opposed to browser-based *ads targeting* where users are tracked and profiled against their visited

10. Google Ad personalisation https://adssettings.google.com/authenticated

websites and ad clicks, the users' interests are based on various norms: selection of mobile apps installed and used, (traditional) browser based profiling, ad clicks of certain categories of ads, demographics e.g. age-rank, location etc.; to create a unified user profile. User profiling and its related *ads targeting* expose sensitive information about users [5], [15], e.g. the target could browse through gambling websites or spends excessive time using gambling apps, revealing (including a third-party, such as the website owner) to the advertising systems that the user is a heavy gambler.

In particular, we address two privacy attacks: *Legitimate user profiling* by Analytics, in traditional advertising systems, these functionalities are implemented via an analytics SDK [32] for reporting user's activities to Analytics & Ads networks, hence, observe various requests to/from mobile users. *Indirect privacy attack*, involves third parties that could intercept and infer user profiles based on *targeted* ads, since ad's traffic is sent in clear text [6]. We presume that users do not wish to be exposed to their private interests and are willing to receive relevant ads based on their interests.

In addition, another problem being addressed where (user) clients or servers outsource their functionalities (e.g. a firewall to monitor incoming/outgoing transactions for authorising various entities to access particular service/resource) to a third party e.g. to cloud system (CS). Hence, in addition to private *ads targeting*, the user wishes to have private outsource functionalities where any third party, including CS, does not learn the *policies* implemented.

2.4 Threat Model

Our privacy goal in this work is to decouple, and hence limit the information leakage to either advertising systems or third party, (mobile) users from (advertising) servers for their interactions, hence, various functionality (i.e. *policies*) are implemented by cloud on their (both user's and server's) behalf. In addition, we limit the information leakage for various other activities e.g., tracking user's for their activities, *billing* for *ad presentation* and *ad click* etc. To an extent, we emulate the "traditional" setting where the mobile user's profiling is entirely implemented on user's side i.e. *Miner*, hence revealing no information of user profiling to advertising systems. The *Miner* requests for corresponding encrypted ads associated with hash of profile's interests. Hence, the CS only learns the match for some unknown (hashed profile's interests) data along with the corresponding other (encrypted ads) data pointers, i.e. any entity in CS is assumed to be an *honest-but-curious*.

3 System Components

Figure 2 shows various components of proposed system.

3.1 Evaluating Advertisements

The Ad Placement Server (APS) groups advertisements according to various interest categories for *targeting* users with specific interests. The advertising system calculates as a similarity match between individual interests and the set of ads; where individual interests and ads are characterised by a set of keywords[11]. The similarity is calculated based on the $tf \cdot idf$ metric [33]. We note that this strategy is not metric specific and that other similarity metrics might be used. Let D is the set of ads from all the advertisers wanting to advertise with an advertising agency. Each ad Ad_{ID} is represented with a unique identifier $ID = 1, ..., D$ and is characterised by $\kappa_{Ad_{ID}}$ keywords. Let the function f returns the set of keywords of an ad i.e. $f\left(Ad_{ID}\right) = \kappa_{Ad_{ID}}$ from the pre-defined list of ads and their corresponding keywords. These set of keywords are then used by another function g for calculating an appropriate interest(s) by calculating the similarity between the ad's keywords $\kappa_{Ad_{ID}}$ and the list of keywords of individual interests i.e. $\kappa_{k,l}$.

$$g\left(\kappa_{Ad_{ID}}\right) = \kappa_{k,l} : sim\left(\kappa_{k,l}, \kappa_{Ad_{ID}}\right) = \bigcup_{\forall Ad'_{ID} \in D} \max\left(sim\left(\kappa_{k,l}, \kappa_{Ad'_{ID}}\right)\right) \tag{7}$$

This similarity $sim\left(\kappa_{k,l}, \kappa_{Ad_{ID}}\right)$ based on $tf \cdot idf$ is calculated as follows: Let t be a particular keyword in $\kappa_{Ad_{ID}}$ and d be the set of keywords in $\kappa_{k,l}$; then the *term frequency* (occurrences of a particular keyword t in d) is $tf_{t,d}$. The *inverse document frequency* of $t \in \kappa_{Ad_{ID}}$ within d is calculated as $idf_t = log\frac{N}{df_t}$, where N is the collection of keywords of interests in *Interest* categories and the df_t is the *document frequency* that is the number of *Interest* categories that contain t. The $tf \cdot idf$ is then calculated as $tf \cdot idf_{t,d} = tf_{t,d} \times idf_t$. Thus the score for $\kappa_{Ad_{ID}}$ can be calculated as:

$$score\left(\kappa_{Ad_{ID}}, d\right) = \sum_{t \in \kappa_{Ad_{ID}}} tf \cdot idf_{t,d} \tag{8}$$

The similarity *score* ranges $0 \leq score \leq 1$ for individual ads, where an ad (or a set of ads) with highest similarity match to profile interests is calculated as a candidate ad for $g_{k,l}$ and is assigned to particular interest e.g. $g_{k,l} \leftarrow Ad_{57}$. Note that a single ad can be assigned to various profiling interests, hence, targeting vast majority of audience. This process is continued for entire set of advertisements. Note [31] that the served ads are either URLs of the sponsoring merchants (advertisers) or they are related to e.g. the Google Play Store apps/games[12] i.e. *self-advertising*. Recall, an app marketplace includes a set of $\mathcal{A}$ mobile apps/games $a_{i,j}$, grouped in different categories, characterised by a set of keywords i.e. $\kappa_{i,j}$. We argue that such advertisements can either be evaluated through the $sim\left(\kappa_{k,l}, \kappa_{i,j}\right)$ similarity metric or a direct match between the profiling *Interests'* Ψ_l and *Apps'* Φ_j categories e.g. using Jaccard Index $\frac{\Psi_l}{\Phi_j}$.

3.2 Hashing Ads and Encrypted Profile Interests

All the advertisements are associated with various profile interests, as mentioned in previous step. The advertising server now evaluates the hashes for all the interests $g_{k,l} : \forall k \in G_l, l \in \epsilon$ under individual categories Ψ_l, i.e. $h\left(g_{k,l}\right)$; in addition, the corresponding ads are encrypted (e.g. using RSA) i.e. $Enc(Ad_{ID})$, we call this dataset a '`profiling-ads`'. Note that this operation is done just

11. The advertising agency and the advertiser agree on various keywords so as to describe advertisements.

12. https://play.google.com/store/apps

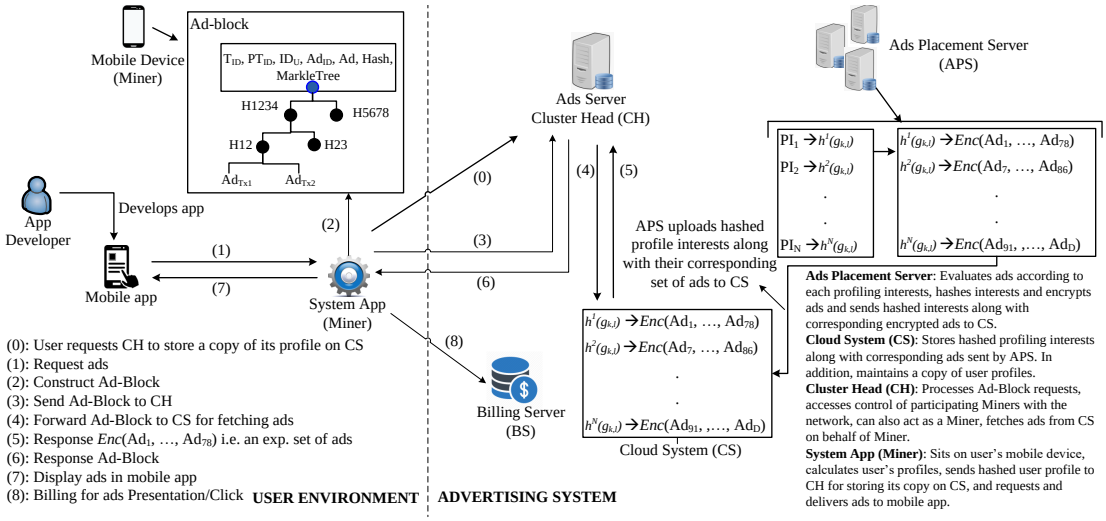


Fig. 2: A framework for secure user profiling and Blockchain-based *targeted* advertising system for *in-app* mobile ads.

once during system initialisation or every time new ads are added to the system for advertising. Each hash references various set of ads e.g. $h\left(g_{k,l}\right) \rightarrow Enc\left(Ad_{ID1}, \ldots, Ad_{IDN}\right)$, which means that a user with specific $h\left(g_{k,l}\right)$ will be *targeted* with $Ad_{ID1}, \ldots, Ad_{IDN}$ ads.

3.3 Storing `profiling-ads` in `CS`

Once the set of profiling interests and their corresponding ads (i.e. `profiling-ads`) are ready, the `APS` can upload it to Cloud Storage (`CS`), which is carried out through an anonymous process similar to those discussed in [34]. Note that `APS` need to go through the *access policy* mechanism to upload (store) the profiling interests and corresponding ads. For this, `APS` sends a request to `CS` for storing `profiling-ads`, requests the pointer to the first block P_b, and the confirmation for availability of storage space. The `CS` evaluates *access policy* for `APS` to confirm whether the request is coming from legitimate party and to allow `APS` to use storage space. Once the `APS` is allowed to store the `profiling-ads` data; the `CS` encrypts the P_b with $PK+$ and sends it to `APS` along with the the `storage-block` B i.e. the capacity of storage block within `CS` for saving information on physical disk for *spanned* organisation of storing data. Following, the `APS` evaluates the blocking factor bfr to calculate the number of block to be occupied by `profiling-ads` and sends it to the `CS`. Subsequent, `APS` decrypts the P_b with $PK-$ to extract the P_b, creates a random ID, and starts sending `profiling-ads` to `CS`. The `CS` calculates the hash of the received data, compares it with the received hash, and (upon its validity) stores the `profiling-ads` data. Since `APS` needs to send (a high volume) data in several transactions; for this, the `CS` sends a new P_b (encrypted with the $PK+$) to `APS`.

3.4 Miner (System App)

The *System App* sits on a user's devices, which performs a range of various tasks: deriving user profiles as outlined in Section 2.2, uploads hashed user profiles to `CS`, requests ads from `CS` through `CH`, and delivers requested ads to mobile apps while fulfilling advertising quota, discussed in detail in Section 4. The *Miner* processes all incoming and outgoing requests using a range of various transaction types, as discussed in Section 4.3. The *Miner* authorises various apps participating in profiling or requesting ads via *access policy* (discussed in Section 3) and authenticates various transaction types. In addition, *Miner* stores the list $PK+$ allocated to each mobile app, which the *Miner* only uses to evaluate the request coming from a legitimate app (see Section 3 for details). Note that, as opposed to legacy approach towards ad request or click [2], [6], [15] i.e. encapsulating app' name; only the app's $PK+$ is encapsulated in *ad-block* that ensures anonymity and privacy. *Miner*'s additional functions include: evaluating *billing* for ads, generating *Genesis* transaction, distributing and updating keys, forming *ad-block*, monitoring user's activity for apps, updating and uploading user profiles on removal of mobile apps.

Note that, in order to calculate user profiles, the *Miner* communicates, similar way as AdMob SDK[13] interacts, with installed mobile apps. We are planning an enhanced version of *System App* for future work aiming at user profiling (e.g. frequency of apps usage) for optimising user targeting.

3.5 Billing

We suggest the use of mining any Cryptocurrencies for our *billing* mechanism. We already proposed a *billing* mechanism within an advertising environment [2], using Priced Symmetric Private Information Retrieval (PSPIR) [35] that is originally proposed for private ecommerce along with polynomial commitments[14] and Zero Knowledge Proofs (ZKP)[15]. However, we suggest the use of Cryptocurrency

13. https://admob.google.com/

14. A polynomial commitment [36] allows constant-sized commitments to polynomials (independent of their degree), where the committed value is an evaluation of this polynomial for specific inputs; consequently, only this value (rather than the polynomial coefficients) will be revealed in the opening phase.

15. ZKP [37] is an interactive protocol that enables one party, the prover, to prove to the verifier that a particular statement is true, while the latter learns no additional information regardless of any a priori knowledge he may posses.

for privacy, secure payment and for compatibility with the underlying proposed advertising system over Blockchain.

To enable private *billing*, we denote the price tags $C_{prs}^{Ad_{ID}}$ and $C_{clk}^{Ad_{ID}}$ respectively for *ad presentation* and *ad click* operations. The *Advertiser* ID_{ID} buys the *airtime* from ad agencies for advertising their ads. To initiate `Billing` transaction for buying *airtime*; the advertiser uses her *private key* ($PK-$) and signs message with the amount of Cryptocurrency (e.g. Bitcoin) and `Billing` server's address, requesting a transaction. The *billing* request transaction is bind with other transactions and is broadcasted to the network, where another *miner* validates the transaction. The process of transaction completes and the `Billing` server receives its portion of Cryptocurrency in her *wallet*. Similarly, the *Miner* initiate `Billing` transaction for ads *presentations*[16] or *clicks* respectively by encoding the $C_{prs}^{Ad_{ID}}$ and $C_{clk}^{Ad_{ID}}$ price tags within `Billing` transaction. Note that the *Miner* keeps track of ads presented to a particular app in *tracking-list*, which is forwarded to `CH` only for those ads that were not presented until the next (e.g. 24hrs) session for the purpose of informing `CS` about the fulfillment of advertising quota. We use the following notations for representing various symbols/entities: price tags $C_{prs}^{Ad_{ID}}$ and $C_{clk}^{Ad_{ID}}$ for ad *presentation* and *click*; various *wallets* i.e. *App Developer*'s $wallet_{ID_{APP}}$, *Advertiser*'s $wallet_{AD_{ID}}$, `Billing` Server's $wallet_{BS}$; and (Bitcoin) addresses, i.e. $Add_{ID_{APP}}, Add_{AD_{ID}}, Add_{BS}$. Future work is planned for integrating complete implementation and applicability of Cryptocurrency in *billing* for advertising system.

3.6 Access Control

The *Miner* implements *access policy* (i.e. *local-policy*), holding an *access control list* that allows apps regarding evaluation of different operations e.g. to check whether an app is allowed to request for ads, *billing* etc. The *policies* are also implemented at `CS` (i.e. *outsource-policy*, the `CS` processes *policies* on behalf of *Miner* i.e. executes *policy* on incoming transaction requests destined for accessing various resources i.e. accessing ads) to allow access to various ads evaluated for various profile interests and to `CH` for processing *ad-block* for requesting ads from `CS` on *Miner*'s behalf. Note that, instead of introducing a *policy header* (as suggested by [34]), we evaluate the access from various options present in the *ad-block* (as shown in Figure 4) for efficient access and reduced overhead.

As shown in Figure 4, we represent the *policies* as binary tree T_p consisting of *left child* $l_i\left(T_p\right)$ and *right child* $r_i\left(T_p\right)$ along with the *matching function* $m\left(l_i\left(T_p\right)\right)$, which is represented as an edge between different (child) nodes and the *action function* $a\left(l_i\left(T_p\right)\right)$. The pair (m, a) is a *Policy*. There is a list of *action functions* A (mainly allow or disallow and route to next level of T_p) and the *matching functions* M i.e. the exact *policy* to implement, such that $a \in A$, $m \in M$. An example set of M that *Miner* implements are: Calculate ads quota, Apps to participate in *billing*, *Profile establishment*, Show ads within app, Apps to change *public key* etc. A simple example *policy* structure is given in Figure 3 (b) implemented at `CS` that consists of four pairs of distinct *policies*: $(m_1, a_1), (m_2, a_2), (m_3, a_3), (m_4, a_4)$. The `CS` use these *policies* to permit *Miner* for requesting various *actions*. Here, m_1 is a *match* for evaluating *Miner* for its permission for requesting *ad-block*, likewise, m_4 for storing profile in `CS`.

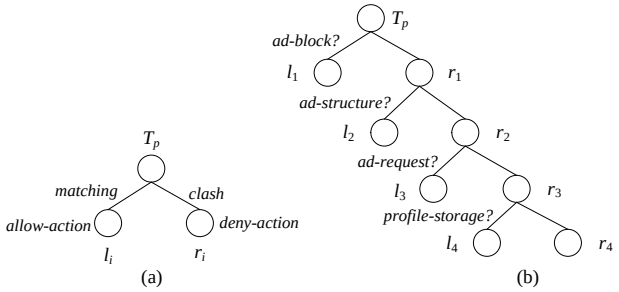


Fig. 3: An example implementation of *policy tree* along with various *actions* (a) and a simple *access policy* implemented at `CS` for *Miner* (b).

We suggest to use Multi-version of *access policy* i.e. creating a separate copy for $read\left(T_p\right)$ and $modify\left(T_p\right)$ *policy document*. This will ensure that the $read\left(T_p\right)$ copy is only accessed for reading purpose, hence, a number of transactions are allowed to access the copy of *access policy*. Note that the *Miner* has to indicate the type of transaction, as mentioned *Trans. Type* in Figure 4, under the `Ad-Block-Structure`.

4 Ad-Block Structure

In this section, we present the block structure for various transactions for an *ad-request/response* for operating within the Blockchain network.

4.1 Ad Block Header Request/Response

Figure 4 presents the structure of a transaction (e.g. an *ad-request*) '`Ad-Block-Header-Request`' (upper) for *ad-request*, '`Ad-Block-Header-Response`' (middle) for *ad-response* and the '`Ad-Block-Structure`' (lower) in a Blockchain network scenario. This block is generated by the *System App*, a *Miner* sitting in user's mobile devices, for requesting an ad (for showing in an app) form `CH`. The T_{ID} denotes the transaction's unique identifier, which is the hashed value of the other fields of this transaction. Note that the first transaction (or block) is called the *Genesis* transaction, which lays the foundation on which additional blocks are sequentially added to form (sequential ledger by connecting current transaction with the previous transaction using PT_{ID}) a chain of blocks. PT_{ID} represents the identifier of the previous transaction that is used for establishing connection between the current and previous transactions. The *input* i.e. all the request ad transactions are made to the *Miner*'s address, enlists all the ads that the `CH` has already sent to the *Miner* within a particular session. Note that both the `CH` and *Miner* keep track of the already served ads so that they are not presented again to user within the same session. We take the session as the time duration where user utilises a particular mobile app until she exits the

16. For reducing processing and communication overhead and enhanced operation, the *Miner* initiates `Billing` transaction once sufficient ads are shown to the user within a session whereas the `Billing` transaction for *click* operation is initiated as soon a *click* operation is triggered.

application or there is no activity for a considerable (e.g. 24hrs) time duration. The *Ad-block* with used "*input*" is then served to a different (*Miner*) user, which upon used by the current *Miner* becomes and stored in the *output* field.

The ID_U[17] represents (user) advertising ID that identifies user's mobile device (detailed discussion over ID_U can be found in [2]). We suggest that ID_U does not change for a particular session. This will help in keeping track of ads shown to the users, in addition to, *billing* where app developer's (ID_{App}) *wallet* is populated with particular amount of Cryptocurrency. Note that for simplicity we represent ID_{App} both for identifying app's developer (Developer ID) and name of the mobile app. The Ad_{ID} is the ID of the advertisement shown to the user. The (mobile) *Miner* also keeps track of particular ads in order to fulfil ad's advertising quota[18]; Table 2 presents an example tracking of advertising quota e.g. the advertisement 'DEF' still needs to be presented 50% app airtime.

Date/Time	Advertisement	Frequency	Served
04/02/2020:10:45:52	ABC	100	65%
05/02/2020:01:23:41	DEF	67	50%

TABLE 2: An example tracking advertising quota.

Furthermore, the Ad is the actual content of the advertisements that are presented to users within mobile apps; the contents of an ad varies over different sizes of *Min* ad size of $12KB$, the *Max* size of $20KB$ and the *Avg* ad of $16KB$, as observed in [2]. Similarly, AD_{ID} is the advertiser's ID of particular advertisement(s), which is used to deduct advertiser's Bitcoin *wallet* (detailed discussion over *billing* is given in 3.5).

To enable communication in a Blockchain network, the peers of a Blockchain network require a *Public-Private Key* pair i.e. represented with $PK+/-$ in Figure 4, which is used to increase the anonymity. The *Sign* field contains the signature of the transaction generator (e.g. *Miner* (*System app*)) using the *public-private key* pair. Following illustrates to understand the exchange of data between *Miner* and `CH` (e.g., *Miner* sends an *ad-request* to `CH`): *Miner* encrypts the *ad-request* transaction data using `CH`'s *public key* $PK+$ and then creates a signature by taking the hash of *ad-request* and encrypts it using her (*Miner*'s) *private key* $PK-$. The transaction data is encrypted along with the digital signature is included in the *Ad-Block-Header-Request/Response*. Subsequent, this transaction is broadcasted over the Blockchain network, since it is addressed[19] to `CH`, hence `CH` starts verifying the transaction contents by decrypting the digital signature using *Miner*'s *public key* while decrypts the transaction data using her (`CH`'s) *private key*. Note that `CH` verifies the transaction data by comparing the *ad- request* hash to the hashed data in digital signature, detailed discussion can be found in [40]. We suggest to, similar to [41], store the *public key* in $PK+/-$ field for reducing the size of the transaction and for future proof transactions i.e. to prevent from malicious attacks where intruders could possibly reconstruct *private key* using *public key*.

The transactions, included in block, are being hashed by the *Miner* as part of the Merkle tree; the transactions stored are lead to the Merkle root stored in the block header. As mentioned in [39], the block header being part of the longest chain with Merkle root is used for different purposes, such as Simple Payment Verification (SPV), and it also speeds up the process of verifying the membership of a transaction in a block. An example Merkle tree, for 8 different *ad-requests* (leaf nodes represented with Ad_{TX1} to Ad_{TX8} on the left), including internal hashed nodes and Merkle root node is shown Figure 5. The hash value of a transaction (e.g. an *ad-request*) is stored in these leaf nodes; note that we presume a binary Merkle tree that has 2^{n-1} leaf nodes with n height. An example slice of a Merkle tree with $n = 8$ is also shown on the right of Figure 5. The yellow nodes are the potential hashed nodes that potentially could be pruned for reclaiming disk space, details for node pruning can be found in [39].

4.2 Ad Block Structure

The lower part of Figure 4 shows the structure of an *Ad-block* '`Ad-Block-Structure`' consisting of the `Ad-Block-Header Request/Response` and the list of some or all recent (pending) *ad-request* transactions. Recall that a *Miner* orders the pending transactions into a block until it reaches the pre-defined block size, as mentioned, blocks in a Blockchain are connected through hashes of previous blocks.

4.3 Ad Block Transaction Types

Various types of communication done among different parties of a Blockchain are enabled via transaction, where different types of transactions are designated for various purposes. E.g. a `Billing` transaction is enabled for calculating *billing* for various ads presented in mobile apps or when the user clicks on particular ads Ad_{ID}, as presented in Section 3.5. Similarly, the `Access`, `Upload`, and `Update` transaction types are respectively used for *accessing* profile[20], *uploading* the profile, or when the user profile *changes*. In addition, the `Request` and `Response` transaction, as discussed above, for requesting ads from `CS` or response ads from `CS`. Furthermore, the `Genesis` transaction is when the *Miner* starts participating in Blockchain network. Note that the `Genesis` transaction is generated for every installed app or for every new app being installed on the device, however, grouped by the app developer ID_{App}. Hence, all the activities i.e. *ad-request*, *ad-response*, *billing* ID_{App} and AD_{ID} both for presented and clicked ads, and participation in profiling are chained together and are tracked for all these activities from current block to the `Genesis` block. In addition, a `Remove` transaction type is used to uninstall an app, indicating that the user might not be interested in this specific category of app and also its contribution to user

17. ID_U can be reset via the Google Settings System App on Android devices

18. This requirement comes from the advertising agency, where a particular is presented for certain frequency [31], [38].

19. The peer's address in a Blockchain network is obtained by calculating the cryptographic hash of user's *public key* e.g. in Bitcoin, the `SHA-256` encryption algorithm is used for deriving user addresses [39].

20. Note that the users upload profiles to `CS`, as explained in Algorithm 4, which can be accessed by the user when she signs-in to another device or losses her profile.

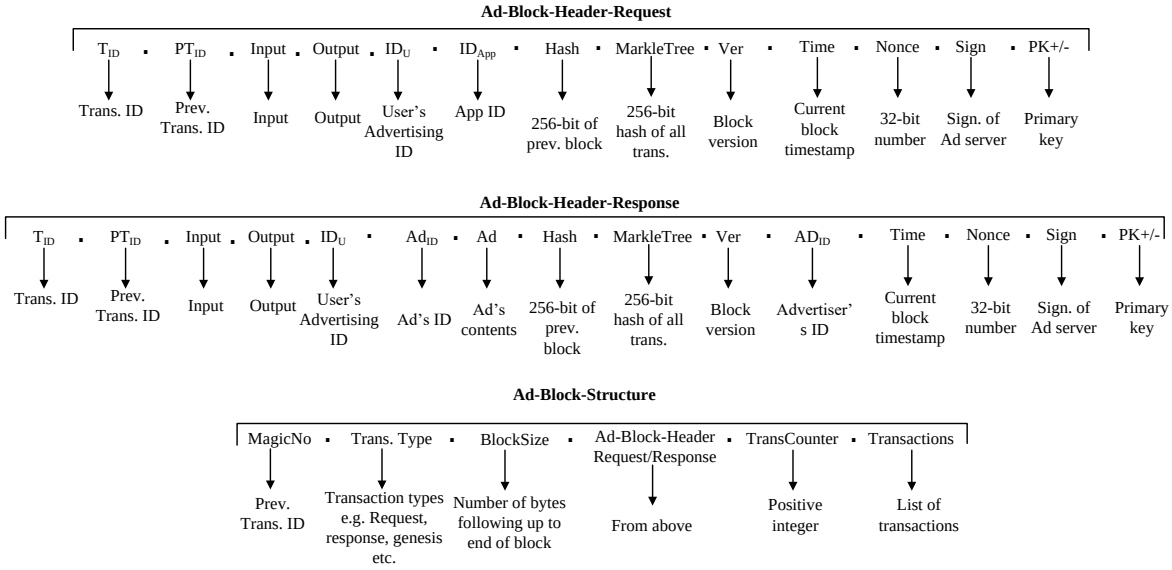


Fig. 4: Block structure of an '`Ad-Block-Header-Request`' (upper), '`Ad-Block-Header-Response`' (middle), and '`Ad-Block-Structure`' (lower) for an *ad-request/response* block in a Blockchain.

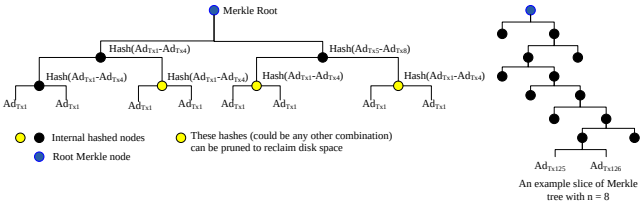


Fig. 5: Ad requests hashed in a Merkle tree (left). A slice of Merkle tree for a height of $n = 8$ (right).

profile. Lastly, the `Monitor` transaction is generated by the *Miner* to monitor activity of each app for various operations of *ad-request*, *ad-response* etc. Note that the transactions are secured via asymmetric encryption, digital signatures and cryptographic hash functions (e.g. `SHA-256`). Furthermore, the lightweight hashing [42] is employed to detect any changes in the transaction's content.

5 Ad-Block Operations

This section presents the design requirements of a Blockchain-based mobile advertising system and the design of various algorithms describing comprehensive operations of its various components.

5.1 Design Requirements

The system is designed by taking into account various requirements: privacy, reduced overhead over (user) client, compatibility of *billing* with Cryptocurrency, efficiency at ad network and client.

Privacy: The underlying design uses Blockchain technology for preserving user privacy, hence various transactions are buried under the structure of Blockchain transaction for *ad-request*, *ad-click*, *billing* etc., hence, the ad network does not learn any information from these various operations. In traditional advertising setting, the `APS` establishes user profiling with the help of cookies [31] and communicates with the mobile devices for various other operations of ad presentation, click etc. However, in proposed method, the *Miner* constructs user profiles locally on the user device[21], hashes of profile interests $g_{k,l}$ and uploads to `CS` or utilise it for various operations including *billing*. The `CS` learns no information about user's interests, either during *policy* checking or accomplishing various operations.

Reduced overhead over (user) client: It is critical to impose marginal overhead over *Miner*, hence, it is made optional for user devices to participate as a *Miner* or request `CH` to construct *ad-request*, henceforth, the *Miner* sends $h\left(ID_U\right)$ and $h\left(ID_{APP}\right)$. However, it is crucial for *Miner* to evaluate various other essential operations (i.e. a trade-off between privacy and operational overheads) in order to enable a private and secure advertising system, e.g., constructing user profile and uploading to `CS`, keep updating the *tracking-list*, *billing* for ads, and initiating the *Genesis* transaction. The aim is to achieve reduced infrastructure cost, efficiency, and flexibility.

Compatibility of billing with Cryptocurrency: The biggest industries, such as travel, banking sectors, are inclined towards adapting the Cryptocurrency that will bring industries, government and community together. In addition, the number of mobile internet users have surpassed 4 billion mark[22] and is set to continue growing in the future. This proposal is an effort to adapt the Cryptocurrency in advertising system for providing a platform for embedding Cryptocurrency for *billing* various parties within the ad system.

Efficiency at ad network and (user) client: The Blockchain based advertising system should be computa-

21. Among other functionality, an important feature of the proposed architecture is to reduce interaction between user device and advertising system, hence the ad system will not be able to learn more about user's activities.

22. https://99firms.com/blog/mobile-marketing-statistics/

tionally fast with reduced computation and, in specific, communication cost. As evident in [31], in traditional *in-app* advertising setting using Google AdMob, a mobile app requesting an ad and the triggered action from e.g. a user click, consists of a series of different steps: The mobile app sends a `POST` message passing various information about the device including phone version, model, app running on phone etc.; the ad server sends ad containing web address, `JavaScript`, static objects; cookie for tracking user's action; the web server associated with the ad is contacted and the landing page is downloaded and is shown to the user. Note that the mobile app developer agree on integrating ads in mobile apps by embedding ad/analytics library. It was also evident that the ads are refreshed every 15–20sec and all the above steps are followed for every single ad presentation and its associated click, hence, severely affecting throughput. This proposal reduces the communication overhead by sending a pool of ads with single *ad-request* transaction and locally serving ads. In addition, the `APS` uploads the `profiling-ads` set only during the system initialisation or when new advertiser is added to the system.

Following illustrates the idea behind various operations of the Blockchain-based advertising system.

5.2 The Blockchain-based Advertising System

The initial step is performed by `APS` via Algorithm 1. This includes evaluating `profiling-ads` and then uploading the profile interests $g_{k,l}$ and corresponding ads Ad_{ID} to `CS`.

Input: $g_{k,l}$; Ad_{ID}; $L_{k,l}$
Run $L_{k,l}$ = `Profiling` ($g_{k,l}$; Ad_{ID})
Run `Storage` ($L_{k,l}$)

Algorithm 1: `Global Setup (APS)`

Once the `APS` completes evaluating `profiling-ads`, it uploads this dataset to `CS` using Algorithm 3 requesting the pointer to the first block P_b for data storage. We assume that the storage in `CS` consists of various blocks of size B i.e. `storage-block`, where each B can accommodate numerous records, hence to evaluate various data records that could be fit in one block. The `APS` sends the *bfr* i.e. Blocking factor, to `CS` to accordingly reserve the storage space for `profiling-ads`. Following, the `APS` starts sending the `profiling-ads` and progressively asks the P_b to next available B.

The `CS` runs the Algorithm 7 for evaluating access permission. To initiate this process, the `CS` creates a $read\,(T_p)$, sets s to the initial node in T_p, and starts traversing through T_p. Recall an example tree-structured *policy* shown in Figure 3, representing $k = 4$ distinct policies and corresponding *matching criteria* m and their resultant *action functions* a. During the traversal, if a *match* is found, the corresponding *action* is performed, however, the *policy traversal* terminates whenever a *leaf* node is entered.

The Algorithm 4 illustrates the procedure for storing user profiles on `CS`, once constructed as a result of *Profile development process*, as explained in Section 2.1. The *Miner* first evaluates the cryptographic hash, using `SHA-256`, of individual profile's interests and then uploads to `CS` using

Input: $g_{k,l}$; Ad_{ID}
Initialise empty list $L_{k,l}$ with $k \times l$ cells.
for $ID = 1, \ldots, D$ **do**
  $\kappa_{Ad_{ID}} \leftarrow f\,(Ad_{ID})$
  for $l = 1, ..., \epsilon$ **do**
    for $k = 1, ..., G_l$ **do**
      Evaluate:
      $core\,(\kappa_{Ad_{ID}}, d) = \sum_{t \in \kappa_{Ad_{ID}}} tf.idf_{t,d}$;
      $\forall Ad_{ID}, \forall g_{k,l}$
    end
    $\bigcup_{\forall Ad_{ID} \in D} \max\left(sim\left(\kappa_{k,l}, \kappa_{Ad'_{ID}}\right)\right)$;
    $L_{k,l} \cdot add\,(g_{k,l}, Ad_{ID})$
  end
end
Evaluate hashes of profile interests and corresponding ads in $L_{k,l}$.
for $m = 1, \ldots, len\,(L_{k,l})$ **do**
  $h^m\,(g_{k,l})\,; Enc_{PK+}\,(Ad_{ID})$
  $L^*_{k,l}.add\,(h^m\,(g_{k,l})\,, Enc_{PK+}\,(Ad_{ID}))$
end
`Return` $\left(L^*_{k,l}\right)$

Algorithm 2: `Profiling`($g_{k,l}$; Ad_{ID})

Input: $L_{k,l}$
`APS` requests `CS` to store $L_{k,l}$
$APS.request\,(P_b)$ from `CS`
if *(Run `Access Policy (APS)`)* **then**
  if *!(storage full)* **then**
    `CS` sends `storage-block` B to `APS`.
    $CS.send\,(Enc_{PK+}\,(P_b))$ to `APS`
  end
end
`APS` sends $bfr = \left\lceil len\,(L_{k,l})/B \right\rceil$
`APS` $Dec_{PK-}\,(P_b)$
`APS` generates random ID
for $i = 1, \ldots, bfr$ **do**
  $APS.send\,(ID, L_{k,l}\,[i]\,, h\,(L_{k,l})\,, Enc_{PK+}\,(P_b))$
  if $h\,(L_{k,l}) = CS.calc\,(h\,(L_{k,l}))$ *AND Transaction is valid* **then**
    $CS.store\,(L_{k,l}\,[i])$
    $CS.send\,(Enc_{PK+}\,(P_b = P_b + 1))$
  end
end

Algorithm 3: `Storage` $(L_{k,l})$, `APS`

Algorithm 3. In addition, the `CS` also evaluates Algorithm 7 to allow/disallow *Miner* ID_U.

The *ad-request* consist of various steps, as outlined in Algorithm 5 and depicted in left side of Figure 2 i.e. *User Environment*. The mobile app ID_{APP} first requests *Miner* for the ads to display to user; Step (1) of Figure 2. The *Miner* also implements an *Access Policy* to evaluate the ad request coming from *legitimate* app. Consequently, the *Miner* constructs an *Ad-Block* using `Ad-Block-Header-Request` by encoding ID_U, ID_{APP} etc., Step (2) of Figure 2. Detailed procedure for constructing `Ad-Block-Header-Request`

Input: $a_{i,j} \in S_a$; $d_{m,n} \in S_d$
Establishment: I_g via use of $a_{i,j} \in S_a$
Map S_a to $S_g^{i,j}$; $I_g \leftarrow S_g^{i,j}$
Evaluate $d_{m,n}$; $I_g \leftarrow d_{m,n}$
Evolution: $I_g^f \leftarrow I_g \cup S'^{i,j}_g \cup d'_{m,n}$
Evaluate hashes of profile interests.
for $m = 1, \ldots, len\left(I_g^f\right)$ **do**
  $h^m\left(g_{k,l}\right)$; $g_{k,l} \in S_g^{i,j} \in I_g^f$
  $I_g^*.add\left(h^m\left(g_{k,l}\right)\right)$
end
Run `Storage`$\left(I_g^*\right)$, `Miner`(ID_U)

Algorithm 4: `Profiling-Storage (Miner)`

is also given in Section 4.1. Following, the *Miner* sends *Ad-Block* to `CS` (Step (3) of Figure 2), which the `CH` forwards to `CS` (Step (4) of Figure 2) for fetching ads according to various $g_{k,l}$ of ID_U. The *ad-response* (Step (5) of Figure 2) is encoded using `Ad-Block-Header-Response` as shown in Figure 4 (middle), illustrated in Section 4.1. Finally, the *Miner* sends the fetched ads to mobile app for display within active session (i.e. Step (7) of Figure 2).

ID_{APP} requests ads from *Miner*
Run `Access Policy` (ID_{APP})
Miner constructs *Ad-Block* using $ID_U, ID_{APP}, \cdots$
Miner sends *Ad-Block* to `CS`
Run `Access Policy` (ID_U)
`CH` forwards request to `CS`
Run `Access Policy` (CH)
Fetch ads according to (ID_U) profile interests from `CS`
Send $h\left(Ad_1, \cdots, Ad_N\right)$ to `CH`
`CH` sends set of ads to *Miner*
Miner presents ads to Mobile app and updates *tracking-list*

Algorithm 5: `Ads-Request (Miner)`

The *Miner* corresponds to `BS` for *billing* for *ad presentation* and *ad click* operations via Algorithm 6, see Section 3.5 for detailed description of *billing*. In order to avoid the communication and computation overhead, we propose to periodically evaluate the *billing* (e.g. every 24hrs) for *ad presentation* through calculating the fulfillment of advertising quota. However, for *click* operation, the *billing* mechanism is initiated as soon as a *click* operation is triggered. Note that u and v are the proportions of *billing* amount (in Cryptocurrency) respectively paid to $wallet_{ID_{APP}}$ and $wallet_{BS}$.

6 Performance Evaluation

We now discuss our experimental evaluation i.e., the processing time for generating *public-private key* $(PK+/-)$ pairs with various key sizes and the encryption and decryption time of advertiser's ads with different number of interests along with its effect over selecting a range of *public-private key* sizes. Note that this wide selection of options affect the *ad presentation* time that include the processing time (including *access policy traversing time*, matching profile hashing at `CS` for fetching relevant ads, the transmission delay (bigger sizes of advertiser's ads are subject to longer transmission

Input: $C_{prs}^{Ad_{ID}}$; $C_{clk}^{Ad_{ID}}$; $wallet_{ID_{APP}}$; $wallet_{AD_{ID}}$; $wallet_{AS}$
Ad Presentation:
Deduct: $B_{prs} = wallet_{AD_{ID}} - C_{prs}^{Ad_{ID}}$
`Billing` transaction to pay *App Developer*: $wallet_{ID_{APP}} \leftarrow \left(B_{prs}/_m\right)$
`Billing` transaction to pay `Billing` server: $wallet_{BS} \leftarrow \left(B_{prs}/_n\right)$
Ad Click:
Deduct: $B_{clk} = wallet_{AD_{ID}} - C_{clk}^{Ad_{ID}}$
`Billing` transaction to pay *App Developer*: $wallet_{ID_{APP}} \leftarrow \left(B_{clk}/_m\right)$
`Billing` transaction to pay `Billing` server: $wallet_{BS} \leftarrow \left(B_{clk}/_n\right)$

Algorithm 6: `Billing` $(ID_{App}, AD_{ID}, Ad_{ID})$

Input: $read\left(T_p\right)$, $modify\left(T_p\right)$, T_p
$s = l_i\left(T_p\right)$
while *traverse until* s *is leaf node* **do**
  if $l_i\left(T_p\right)$ *is not leaf* **then**
    $l_i\left(T_p\right) = 1$
    if *access is allowed* **then**
      Move to the left child.
      $s = l_i\left(T_p\right)$
      Evaluate (m_i, a_i)
    else
      Move to the right child.
      $s = r_i\left(T_p\right)$
    end
  else
    if $l_i\left(T_p\right)$ *is leaf node* **then**
      Move to the right child.
      $s = r_i\left(T_p\right)$
      Exit
    end
  end
end

Algorithm 7: `Access Policy` $(Node)$

delays), which in turn dominates the overall communication delay i.e. the response back from `CS` to *Miner* for *ad request*. Following, to analyze the scalability of the our proposed framework, we evaluate the processing delays for varying number of traversed rules and corresponding actions, represented with different shapes of *access policy trees*.

6.1 Experimental setup

Evaluations were performed on Intel(R) Core(TM) i7-2620M CPU @2.70GHz and 8GB or RAM; we use a `Python` module `cryptography.hazmat.primitives.asymmetric` from `RSA` for generating $PK+/-$ key pairs and for evaluating `AES` encryption/decryption time of advertiser's ads with different sizes of ad's contents. We generate 10,000 $PK+/-$ key pairs with a range of selection of various key sizes (i.e. $512, 1024, 2048$, and 4096 bits) (i.e., we stress-test the proposed framework with continuously generating

a total of 40,000 $PK+/-$ key pairs) and evaluate the key pair generation time. Note that the $PK+/-$ key pairs are generated once and are used for various other operations.

Following, we generate 1000 user profiles by randomly selecting up to 20 different profile interests[23] representing users with diverse interests, equally divide them into 10 groups consisting of 100 user profiles (i.e. $P_1, P_2, \ldots, P_{10}$). Figure 9(a) shows user profile groups uniformly distributed from 1000 user profiles that consist of various profile interests with 10.53 Avg interests with a $St.Dev$ of 5.88 interests. We calculate the hashes of these profiles using `SHA1`, `SHA224`, `SHA256`, `SHA384`, and `SHA512` hashing schemes.

Furthermore, we generate 1000 advertiser's ads with random sizes of between $12KB$ to $20KB$[24] and equally divide them into 10 groups consisting of 100 randomly chosen ads (i.e. $A_1, A_2, \ldots, A_{10}$). Figure 6 shows advertiser's ads groups uniformly selected from 1000 ads with various sizes, with an Avg size of 15943B (i.e. 15.94KB) and a $St.Dev$ of 2364B. Following, we evaluate the `AES` encryption/decryption time of these groups of ads with various key sizes of $512, 1024, 2048,$ and 4096 bits. Note that in order to encrypt ads of arbitrary length, we use hybrid encryption i.e. `RSA` to asymmetrically encrypt a symmetric key. To do this, we first randomly generate a symmetric encryption key using `AES` and encrypt the ads, then, we encrypt the `AES` key using $PK+$ generated with `RSA` and transmit both the `AES`-encrypted ads and `RSA`-encrypted `AES` key to the *Miner*. Following, to decrypt the ad's contents, the *Miner* first decrypts the `RSA` block using $PK+$ to discover the `AES` symmetric key and then decrypt the symmetrically encrypted ads using `AES` to discover the resultant set of ads. Similarly, we construct the `Ad-Block-Structure` for *ad-request/response* by including `Ad-Block-Header-Request` or `Ad-Block-Header-Response` (by encoding $Min = 12KB, Max = 20KB$ sizes of ad' content [2]) respectively for requesting data for *ad presentation* by the *Miner* and the response back from `CS` to *Miner*.

6.2 Beskpoke Miner

We develop a bespoke version of the *Miner*, we made significant changes to implement some of its functionalities, such as generating an *ad-block*, transactions with different *policy* requests etc., on an Android-based device. We implement *Miner* on Google Nexus 7, ARM Cortex-A9 Nvidia Tegra, 1GB RAM, 1.2 GHz quad-core. To implement various functionalities, first we install CyanogenMod that is a custom ROM for Android devices in order to get `root access` and `developer` capabilities. We then add support for the standard Linux utilities, in order to implement various *Miner* functionalities, that are not generally ported as part of the Android OS. The Android OS is equipped with *Bionic* C library[25] instead of Glibc[26] i.e. the standard GNU C library, found on most Unix variant operating systems, which

23. Note that apps generate a very limited number of profile interests, as evident in [5].

24. The selection of these sizes of ads are based on our prior works, which has characterized *in-app* mobile ads [5], [31].

25. https://android.googlesource.com/platform/bionic/

26. http://www.gnu.org/software/libc/

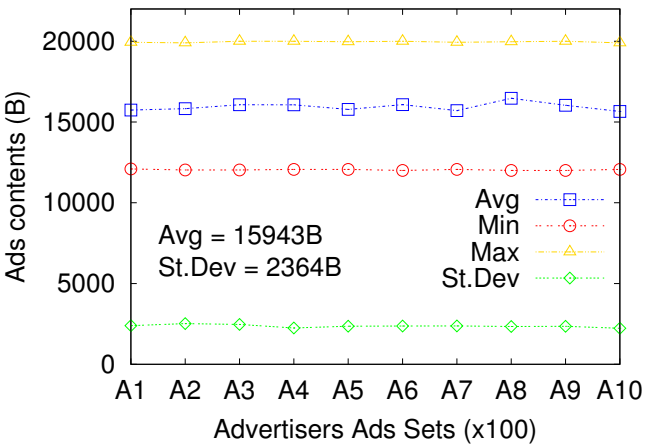


Fig. 6: Randomly generated ads with $Min = 12KB, Max = 20KB, Avg = 15.94KB, St.Dev = 2.364KB$ content's sizes. $A_i = 100\,ads; i = 1, \ldots, 10$.

enable Android OS to achieve high performing standard C library with small memory and disk footprint. Following, we make both the *Glibc* and *Bionic* coexist on Android platform using the `chroot` system call in order to use Linux's utilities; this step also enables us to create Ubuntu Linux file system, which we can run a complete Debian environment on top of the Android kernel ARM CPU support OS. These various layers of functionalities are presented in Figure 7.

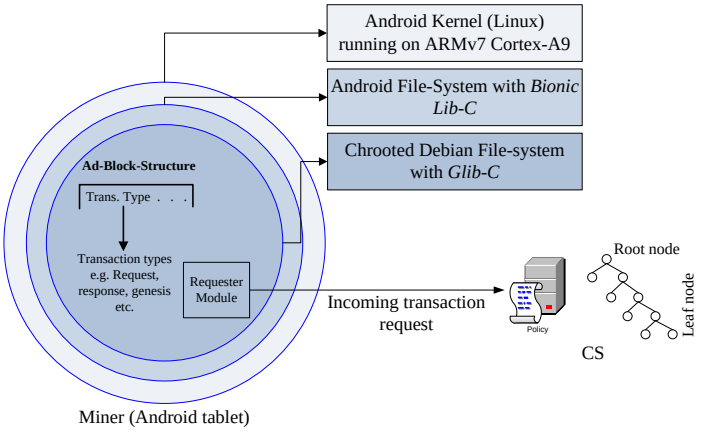


Fig. 7: Runtime architecture for a bespoke *Miner*.

To implement a system module of *access policy* and to evaluate the system performance under different shapes of access trees, we set our `CS` to store $100, 200, \ldots, 1000$ set of rules and matches with root node at *random* positions between $min = 1$ to $max = n$, $n = \{100, 200, \ldots, 1000\}$ representing various shapes of trees with nodes (i.e. *matching rules* and corresponding *actions*) present on both sides. E.g. for a tree of 1000 nodes with root node set to 1 or 1000 means that the entire set of *matching rules* and corresponding *actions* are present respectively on the right and left side of the root node, similarly, the set of rules and matches are equally distributed on both sides with the position of root node at position 500. We repeat this experiment for 1000

times for each of various trees of $100, 200, \ldots, 1000$ nodes with *randomly* selecting root nodes at different positions. Note that for each request, in order to *stress test* the system, we deliberately put the *matching rule* for every *access policy* request at the leaf node of any specific tree, hence traversing the entire tree for each matching rules. We configure the `Requester module` (i.e. inside the *Miner* as depicted in Figure 7) to launch these requests, with above setting, to `CS` (we experiment our `CS` with the same specs as presented in Section 6.1), as shown at the bottom of Figure 7. The processing delay is evaluated at `CS` for all these experiments, along with $Avg, Min, Max, St.Dev$ of root node placement positions and processing delays.

6.3 Experimental results

Processing delay for generating $PK+/-$: Our first evaluation is to explore the effect of $PK+/-$ key pair size over the key pair generation time. Figure 8 shows the performance in terms of processing delays evaluate to $Avg, Min, Max, St.Dev$ for generating $PK+/-$ key pair with various sizes of $512, 1024, 2048$, and 4096 bits. E.g. the Min and Max processing time for generating 10,000 keys with key size of 512 bits respectively is 0.005sec and 0.35sec with an Avg delay of 0.05sec and 0.03 $St.Dev$. We note that the Avg *power trendline* results in a curve with 0.002sec slope and a 0.86 of $r-squared$ value, as the key size is increased from 512 to 4096 bits. The Avg key generation delay is also written on top of each $PK+/-$ key pairs with various key sizes.

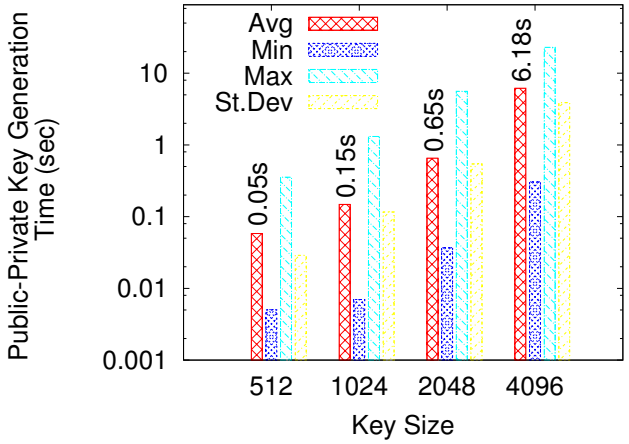


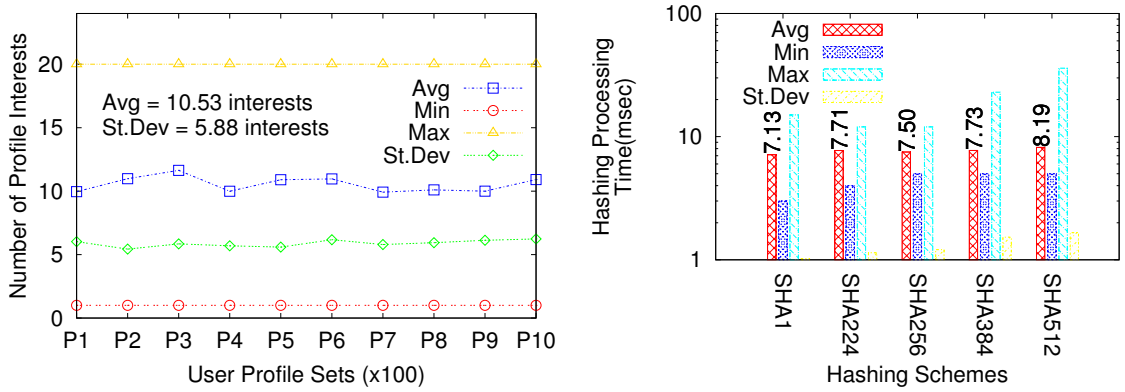
Fig. 8: Processing delay for generating 40,000 $PK+/-$ key pairs with various key sizes of $512, 1024, 2048$, and 4096 bits.

Processing time of Hashing of user profiles: Figure 9(b) shows the processing time of hashing various user profiles with `SHA1`, `SHA224`, `SHA256`, `SHA384`, `SHA512` hashing schemes, showing the $Avg, Min, Max, St.Dev$ processing time (msec) of $P_1, P_2, \ldots, P_{10}$ profile groups with different hashing schemes. For instance, the Min and Max hashing time with `SHA1` are respectively 3msec and 15msec with an Avg and $St.Dev$ of 7.13msec and 1.02ms, while these statistics respectively increase to 5msec, 36msec, 8.19msec and 1.66msec using `SHA512` hashing scheme. The Avg processing time with each scheme is also shown on top of each schemes. Note that this is a one time operation, which `APS` calculates before calculating the profile hashing and their relevant ads before `profiling-ads` to `CS`.

Enryption/decryption time of Advertiser's ads: Figure 10 shows the `AES` encryption (above row from (a) to (d)) and `AES` decryption (below row from (e) to (h)) time of various set of ads with different $PK+$ key sizes i.e. $1024, 2048, 4096, 8192$ bits. The averages of $Avg, Min, Max, St.Dev$ processing time duration of $A_1, A_2, \ldots, A_{10}$ advertiser's ads groups is also shown within each figure for both ad's encryption time (sec) and decryption time (sec) with each evaluation. For instance, the Min and Max ad's encryption time with 1024 bits key size are respectively 0.0010sec and 0.0045sec with an Avg and $St.Dev$ of 0.0022sec 0.0007sec; on the other hand, with 8192 bits of key size, these statistics increase to 0.039sec and 0.0267sec respectively for Min and Max encryption time with Avg and $St.Dev$ of 0.0112msec and 0.0046sec. Note that `APS` needs to encrypt ads only once i.e., at the time of system initialization, as described in Algorithm 2, before uploading to `CS`. Similarly, the decryption time (below row of Figure 10) with Min and Max respectively evaluates to 0.0029sec and 0.0097sec with Avg and $St.Dev$ of 0.0047sec and 0.0013sec with 1024 bits of $PK+$ key size; on the other hand, with 8192 bits of key size, these statistics evaluate to 0.4520sec, 1.6588sec, 1.0819sec, and 0.3232sec respectively for Min, Max, Avg and $St.Dev$.

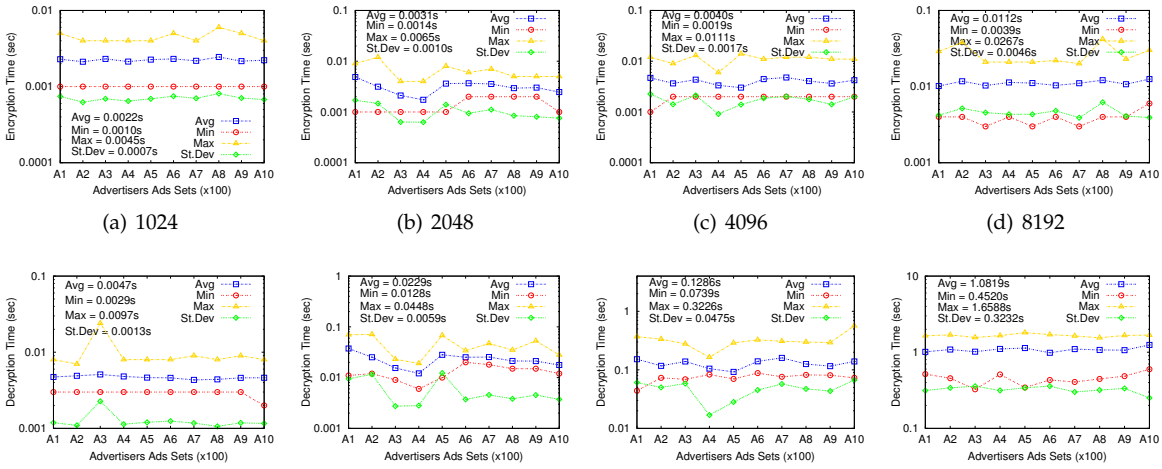
Various statistics of $Avg, Min, Max, St.Dev$ of `AES` encryption and description processing time evaluated from 1000 advertiser's ads with various $PK+$ sizes of $1024, 2048, 4096, 8192$ bits is presented in Table 3. Furthermore, the $r-squred$ values calculated using *exponential trendline* R^2-Exp and power R^2-Pow are also presented that shows the relationship of increasing key size to the encryption/decryption processing time of advertiser's ads. We note that there is a strong correlation between the $PK+$ key size and the ads' encryption or decryption processing time. For instance, Avg `AES` encryption processing time, the R^2-Exp and power R^2-Pow evaluated respectively to 0.8921 and 0.7663 (which also evident with high values of $St.Dev$ for both R^2-Exp and R^2-Pow) showing that the encryption time exponentially increases as the key size is increased from 1024 to 8192.

Processing delay for traversing policies: To further analyze the scalability of the our proposed framework, we study the processing delay (note that this contributes to the total delay of various transactions requested by *Miner* or any other entity that requests `CS` for various services) of $Avg, Min, Max, St.Dev$ when varying the number of traversed rules among $100, 200, \ldots, 1000$ *access policy trees*, for both *sequential* and *random* distribution of *policy tree* i.e. *sequential/random* root node placement. Figure 11 (b) shows the *random* placement of root node for *policy trees*, storing various nodes and matching rules of $100, 200, \ldots, 1000$ access *policy trees*. We note that the root node placement, on average, perfectly follows a power distribution with R^2-Pow value of 0.99, as the tree is populated with 100 to 1000 policies and matching rules. Similarly, the R^2-Pow for $Max, St.Dev$ is also 0.99, suggesting formation of various



(a) User profiles (b) Hashing

Fig. 9: Randomly generated user profile (a) with $Min = 1, Max = 20, Avg = 10.53, St.Dev = 5.88$ profile interests. $P_i = 100\, user\, profiles$; $i = 1, \ldots, 10$. Processing time of hashing profiles (b) using `SHA1`, `SHA224`, `SHA256`, `SHA384`, `SHA512` hashing schemes.



(a) 1024 (b) 2048 (c) 4096 (d) 8192

(e) 1024 (f) 2048 (g) 4096 (h) 8192

Fig. 10: `AES` Encryption time (a) to (d) and Decryption time (e) to (h) using $1024, 2048, 4096, 8192$ bits of `RSA` $PK+$ key, evaluated for $A_1, A_2, \ldots, A_{10}$ advertiser's ads groups.

access policy with placement of root nodes at every possible positions i.e., creating every possible shape of tree by placing nodes on both sides of the access *policy trees*.

Figure 11 (a) shows processing delay of $Avg, Min, Max, St.Dev$ *access policy* traversal delay (msec) with various access trees; note that every experiment has been repeated 1000 times. We observe that the average processing delay increases with the number of rules with a *power trendline* of 0.92. For instance, for a 100 and 1000 node's *policy tree*, the Avg processing delay is respectively 0.0017sec and 0.0106sec, similarly, the Min processing delay varies respectively from 0.001sec to 0.005sec. Detailed statistics of root node placement and traversal delay is also given in Table 4, evaluated for $Avg, Min, Max, St.Dev$ processing delays.

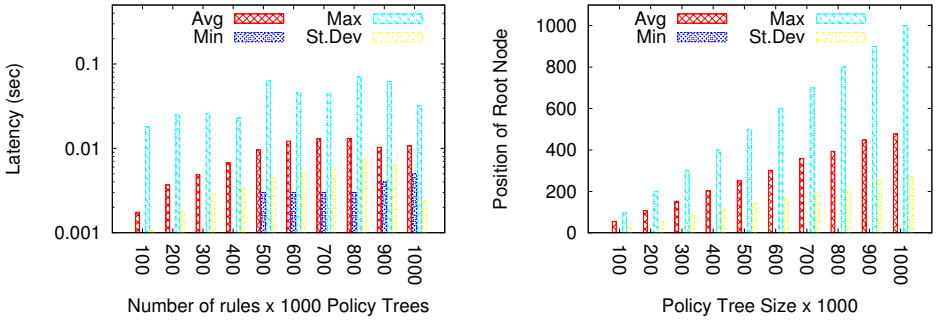
Following, we evaluate the processing delay using *sequential* (distribution) placement of root node (i.e. to evaluate traversal delay by placing root node at every single position e.g. $1, 2, 3, \ldots, 1000$ for a tree of 1000 matching rules and their actions) and compare it with the *random* distribution. Figure 12 presents a CDF of processing delay with both *random* and *sequential* placement of rood node; various processing delays of $Avg, Min, Max, St.Dev$ is also given at the right side of this figure. We note that 96% of the processing delay ranges from 0.005sec to 0.014sec and 0.007sec to 0.032sec respectively for *random* and *sequential* distribution of root node placement. Overall, the *sequential* distribution resulted in higher processing delays as compared to *random* distribution. Note that these processing delay at `CS` can be considered acceptable i.e., close to real time performance [2], [38], both during the app/website *launch time* or *delayed* ad request.

7 Related Work

Privacy threats resulting from collecting individual's online data i.e. direct and indirect (inferred) leakage, have been investigated extensively in literature, e.g. [1], [43], [44], [45],

Stats	Encryption Time (sec)						Decryption Time (sec)					
	Key Size (bits)				Trendline		Key Size (bits)				Trendline	
	1024	2048	4096	8192	R^2-Exp	R^2-Pow	1024	2048	4096	8192	R^2-Exp	R^2-Pow
Avg	0.0022	0.0031	0.0040	0.0112	0.8921	0.7663	0.0047	0.0229	0.1286	1.0819	0.9953	0.9320
Min	0.0010	0.0010	0.0010	0.0030	0.6000	0.4307	0.0020	0.0060	0.0440	0.3260	0.9839	0.8984
Max	0.0060	0.0120	0.0140	0.0420	0.9223	0.8496	0.0240	0.0710	0.5670	1.8070	0.9840	0.9323
St.Dev	0.0007	0.0014	0.0018	0.0046	0.9627	0.9008	0.0013	0.0097	0.0534	0.3308	0.9992	0.9690

TABLE 3: The $Avg, Min, Max, St.Dev$ of AES encryption and description processing time evaluated from 1000 advertiser's ads of various sizes with different $PK+$ sizes of $1024, 2048, 4096, 8192$ bits.



(a) Processing delay (ms)

(b) Root node selection

Fig. 11: $Avg, Min, Max, St.Dev$ Processing delay (ms) per *policy tree* vs number of rules traversed (x1000) for each *policy matching* requested by *Miner* (a) and root node placement for each set of *policy trees* (b).

Tree Size	Policy Tree Processing Delay (sec)				Policy Tree - Rood Node Selection			
	Avg	Min	Max	St.Dev	Avg	Min	Max	St.Dev
100	0.0017	0.001	0.018	0.001	56.08	1	100	26.95
200	0.0037	0.001	0.025	0.002	106.63		200	53.63
300	0.0049	0.001	0.026	0.003	152.98		300	84.94
400	0.0067	0.001	0.023	0.003	202.37		400	115.29
500	0.0096	0.003	0.063	0.004	250.99		500	141.94
600	0.0122	0.003	0.046	0.005	301.90		600	166.34
700	0.0131	0.003	0.045	0.006	359.22		700	196.66
800	0.0131	0.003	0.071	0.007	393.72		800	224.08
900	0.0103	0.004	0.062	0.006	447.85		900	255.95
1000	0.0107	0.005	0.032	0.002	477.54		1000	271.96
$R^2 - Pow$	**0.9166**	**0.7491**	**0.5778**	**0.6673**	**0.9994**	**0.0436**	**0.9996**	**0.9991**
$R^2 - Exp$	**0.7196**	**0.8497**	**0.4804**	**0.4553**	**0.9084**	**0.0303**	**0.8985**	**0.8940**

TABLE 4: $Avg, Min, Max, St.Dev$ *Policy tree* processing delay (sec) (left) and the *random* root node placement position (right) for *policy trees* of $100, 200, \ldots, 1000$; the *exponential trendline* $R^2 - Exp$ and power $R^2 - Pow$ are also presented. Each statistic is a result of 1000 experimental request.

including third party ad tracking and visiting [46], [47], [48], [49]. These studies have shown the prevalence of tracking on both web and mobile environments and demonstrate the possibility of inferring user's PII, such as age, gender, relationship status, email address, etc. from their online data. The ad and analytics libraries, integrated into mobile apps, systematically collect personal information and users' *in-app* activities and are more likely to leak this information to ad systems and third party tracking. The authors in [50] analyze the privacy policy for collecting *in-app* data by apps and study various information collected by the analytics libraries integrated in mobile apps.

Privacy and mobile advertising: There are number of studies that focus on privacy leakage caused by ad libraries within the mobile apps, e.g. the authors in [51] show the potential privacy and security risk by analyzing 100,000 android apps and find that majority of the embedded libraries collect private information. Another study [52] finds that majority of the used APIs within mobile apps do not implement private APIs and send private information to ad servers. The authors in [53] manually create user profiles and find that ads are vastly targeted based on fabricated profiles, in addition to, apps used and location. In our previous study [31] and in [38], the authors study various information sent to ad networks and the level of ads targeting based on communicated information, similarly, [54] investigates installed apps for leaking targeted user data. The authors in [55] studies the privacy risks collected by ad libraries across various apps installed on a single device called intra-library collusion; the longitudinal analysis shows that leakage of information has increased in the following two years. Furthermore, [45] studies the level of information collected by the ad networks with the help of installed and used apps, in addition, to reverse engineer the targeted ads and investigate the collected information.

Privacy and mobile Analytics services: In our previous work [6], we study the user's sensitive information leakage through the vulnerabilities in mobile analytics services by analyzing actual network traffic generated by mobile apps and reconstruct the exact user profiles of a number

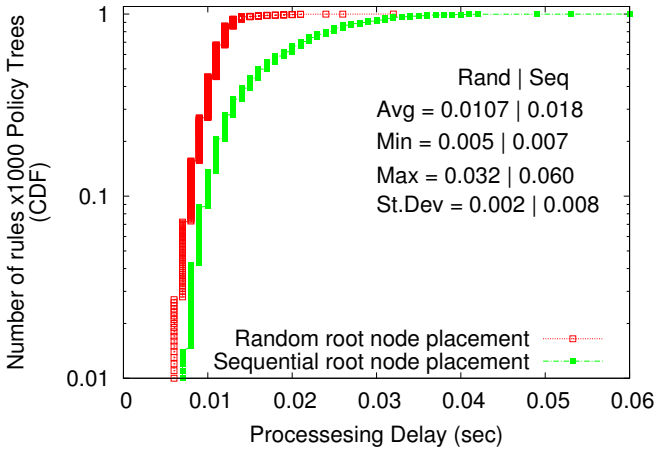


Fig. 12: $Avg, Min, Max, St.Dev$ Processing delay (ms) per *policy tree* vs number of rules traversed for rules matching requested by *Miner* for *Random* and *Sequential* root nodes placement.

of participating volunteers. We mainly carry out this study on android platform for Google Mobile Analytics and Flurry Analytics. We further study how the ads served to users can be influenced by modifying the user profiles generated by these analytics services. Another study [56] studies third-party analytics and ad libraries, and find that 60% of the paid apps contain at least one tracker that collects personal information. A similar study [32] examines how users are tracked by running android apps; they find that 57% of the apps and every participant in their study is tracked several times for sensitive information. The authors in [57] collects network traffic using ' ICSI Haystack' app and distinguishes between ad and analytics libraries and for collecting various information. Furthermore, the authors in [58], [59] study the distribution of third-party trackers across web and android apps and their impact on user privacy.

Analysis of third-party tracking scripts: Third-party web tracking, the practice using which websites identify and collect information about users, has been widely studied in literature, e.g. [60] develops a client side and identify 500 unique trackers and finds that majority of commercial websites are tracked by multiple parties. The authors in [61] develop a tool TrackingExcavator' for longitudinal measurements of third-party web tracking 1996-2016 and finds that tracking mechanism is constantly becoming complex and widely used. An open-source[27] privacy measurement tool 'OpenWPM' automatically detects and characterizes emerging online tracking behaviors. The 'TrackingFree' [62] provides an anti-tracking browser solution that automatically breaks the tracking chain for third-party tracking web tracking by disabling unique identifies. Similarly, [48] analyze the protection level offered across a wide range of web and mobile apps and evaluates their effectiveness for blocking third-party tracking. They find that more than 60% of third-party tracking services communicate in plain text.

Privacy-preserving mobile advertising: There are a number of privacy-preserving personalization and architectures both for in-browser and *in-app* that propose private profiling and advertising systems, such as, Adnostic [20], Privad [19], RePriv [63], MobiAd [64], Splitx [65]. Adnostic, Privad and RePriv are browser-based extensions that focus on targeting users based on their browsing activities and locally carry out profiling (i.e. in the user's browser). MobiAd suggest local user profiling and receives ads in Delay Tolerant Networks (DTN) fashion via broadcasting stations and WiFi hotspots. Similarly, SplixX uses differentially private queries over distributed user data. Furthermore, another noisy technique 'Bloom cookies' [66] presents a framework for privacy preserving personalization of web searches of locally derived profile. In our previous works, we propose 'ProfileGuard' [5], [15] that is an app-based profile obfuscation mechanism and a privacy-preserving mobile advertising system based on various implementation of Private Information Retrieval (PIR) [2].

The authors in [21] present a targeted mobile coupon delivery scheme based on Blockchain, however this framework does not include all components of the advertising system including user profile construction, detailed structure of various transactions and operations, or other entities from our work, such as the *Miner* and the *billing* process. In addition, it does not specify the technical details of *access policy* to control access to resources, e.g., ads. Another work [67] implements a protocol for personal data management using Blockchain that ensures user's ownership and control over data. We present a decentralized Blockchain-based advertising system that ensures private user profiling, in addition, the transfer of information between user and `CS` and between `CS` and `APS` uses an encryption algorithm to ensure data security. Furthermore, the control of information and user's ownership and control over its profiling data is ensured using *access policy*. We also propose private *billing* mechanism for ad presentations and clicks based on Cryptocurrency. For scalability of the system, we introduce the cluster head that can process *ad-block* requests for *Miner* for storing user profiles and fetching ads from `CS`.

8 Conclusion

The *in-app* mobile advertising is growing in popularity, while the advertising companies use excellent user-tracking tools, which have raised concerns among privacy advocates. This paper presents the design of a Blockchain-based framework for advertising platform in addition to implementation of its various critical components that allows the cloud system to privately compute access policies both for mobile users and ad systems for accessing various resources, private user profiling, private *billing*, privately requesting ads from the advertising system based on user interests, and devising various transactions for enabling advertising operations in Blockchain. We develop POC implementation of our proposed framework and perform a thorough system evaluation and stress test its various components in order to evaluate different constraints of processing delays. Our evaluations show an acceptable amount of processing delay for performing various operations of accessing relevant ads as that of the currently implemented ad system in an actual environment.

27. https://github.com/mozilla/OpenWPM

REFERENCES

[1] S. Mamais, *Privacy-preserving and fraud-resistant targeted advertising for mobile devices*. PhD thesis, Cardiff University, 2019.

[2] I. Ullah, B. G. Sarwar, R. Boreli, S. S. Kanhere, S. Katzenbeisser, and M. Hollick, "Enabling privacy preserving mobile advertising via private information retrieval," in *2017 IEEE 42nd Conference on Local Computer Networks (LCN)*, pp. 347–355, IEEE, 2017.

[3] Y. Liu and A. Simpson, "Privacy-preserving targeted mobile advertising: requirements, design and a prototype implementation," *Software: Practice and Experience*, vol. 46, no. 12, pp. 1657–1684, 2016.

[4] Y. Wang, E. Genc, and G. Peng, "Aiming the mobile targets in a cross-cultural context: Effects of trust, privacy concerns, and attitude," *International Journal of Human–Computer Interaction*, vol. 36, no. 3, pp. 227–238, 2020.

[5] I. Ullah, R. Boreli, S. S. Kanhere, S. Chawla, T. A. Ahanger, and U. Tariq, "Protecting private attributes in app based mobile user profiling," *IEEE Access*, vol. 8, pp. 143818–143836, 2020.

[6] T. Chen, I. Ullah, M. A. Kaafar, and R. Boreli, "Information leakage through mobile analytics services," in *15th International Workshop on Mobile Computing Systems and Applications*, ACM HotMobile, 2014.

[7] M. M. Tsang, S.-C. Ho, and T.-P. Liang, "Consumer attitudes toward mobile advertising: An empirical study," *International journal of electronic commerce*, vol. 8, no. 3, pp. 65–78, 2004.

[8] M. Merisavo, S. Kajalo, H. Karjaluoto, V. Virtanen, S. Salmenkivi, M. Raulas, and M. Leppäniemi, "An empirical study of the drivers of consumer acceptance of mobile advertising," *Journal of interactive advertising*, vol. 7, no. 2, pp. 41–50, 2007.

[9] G. A. Johnson, S. K. Shriver, and S. Du, "Consumer privacy choice in online advertising: Who opts out and at what cost to industry?," *Marketing Science*, 2020.

[10] A. Razaghpanah, R. Nithyanand, N. Vallina-Rodriguez, S. Sundaresan, M. Allman, C. Kreibich, and P. Gill, "Apps, trackers, privacy, and regulators: A global study of the mobile tracking ecosystem," 2018.

[11] M. Elsabagh, R. Johnson, A. Stavrou, C. Zuo, Q. Zhao, and Z. Lin, "{FIRMSCOPE}: Automatic uncovering of privilege-escalation vulnerabilities in pre-installed apps in android firmware," in *29th {USENIX} Security Symposium ({USENIX} Security 20)*, 2020.

[12] J. Ren, A. Rao, M. Lindorfer, A. Legout, and D. Choffnes, "Recon: Revealing and controlling pii leaks in mobile network traffic," in *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, pp. 361–374, 2016.

[13] L. Verderame, D. Caputo, A. Romdhana, and A. Merlo, "On the (un) reliability of privacy policies in android apps," *arXiv preprint arXiv:2004.08559*, 2020.

[14] D. Wermke, N. Huaman, Y. Acar, B. Reaves, P. Traynor, and S. Fahl, "A large scale investigation of obfuscation use in google play," in *Proceedings of the 34th Annual Computer Security Applications Conference*, pp. 222–235, 2018.

[15] I. Ullah, R. Boreli, S. S. Kanhere, and S. Chawla, "Profileguard: Privacy preserving obfuscation for mobile user profiles," pp. 83–92, 2014.

[16] R. Balebako and L. Cranor, "Improving app privacy: Nudging app developers to protect user privacy," *IEEE Security & Privacy*, vol. 12, no. 4, pp. 55–58, 2014.

[17] R. Balebako, A. Marsh, J. Lin, J. I. Hong, and L. F. Cranor, "The privacy and security behaviors of smartphone app developers," 2014.

[18] S. Jain, J. Lindqvist, *et al.*, "Should i protect you? understanding developers' behavior to privacy-preserving apis," in *Workshop on Usable Security (USEC'14)*, Citeseer, 2014.

[19] S. Guha, B. Cheng, and P. Francis, "Privad: Practical privacy in online advertising," in *USENIX conference on Networked systems design and implementation*, pp. 169–182, 2011.

[20] V. Toubiana, A. Narayanan, D. Boneh, H. Nissenbaum, and S. Barocas, "Adnostic: Privacy preserving targeted advertising," in *Proceedings Network and Distributed System Symposium*, 2010.

[21] Y. Gu, X. Gui, P. Xu, R. Gui, Y. Zhao, and W. Liu, "A secure and targeted mobile coupon delivery scheme using blockchain," in *International Conference on Algorithms and Architectures for Parallel Processing*, pp. 538–548, Springer, 2018.

[22] K. Kenthapadi, T. T. L. Tran, M. Dietz, T. Greason, and I. V. Koeppe, "Random noise based privacy mechanism," Mar. 14 2019. US Patent App. 15/703,834.

[23] G. Beigi, R. Guo, A. Nou, Y. Zhang, and H. Liu, "Protecting user privacy: An approach for untraceable web browsing history and unambiguous user profiles," in *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pp. 213–221, 2019.

[24] O. Starov and N. Nikiforakis, "Privacymeter: Designing and developing a privacy-preserving browser extension," in *International Symposium on Engineering Secure Software and Systems*, pp. 77–95, Springer, 2018.

[25] N. Laoutaris and J. Blackburn, "Method, a device and computer program products for protecting privacy of users from web-trackers," Oct. 23 2018. US Patent 10,110,633.

[26] O. Rafieian and H. Yoganarasimhan, "Targeting and privacy in mobile advertising," *Available at SSRN 3163806*, 2020.

[27] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts," in *2016 IEEE symposium on security and privacy (SP)*, pp. 839–858, IEEE, 2016.

[28] M. Vukolić, "The quest for scalable blockchain fabric: Proof-of-work vs. bft replication," in *International workshop on open problems in network security*, pp. 112–125, Springer, 2015.

[29] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[30] A. Dorri, M. Steger, S. S. Kanhere, and R. Jurdak, "Blockchain: A distributed solution to automotive security and privacy," *IEEE Communications Magazine*, vol. 55, no. 12, pp. 119–125, 2017.

[31] I. Ullah, R. Boreli, M. A. Kaafar, and S. S. Kanhere, "Characterising user targeting for in-app mobile ads," in *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 547–552, IEEE, 2014.

[32] S. Han, J. Jung, and D. Wetherall, "A study of third-party tracking by mobile apps in the wild," *Technica l Report, ftp://ftp. cs. washington. edu/tr/2012/03/UW-CSE-12-03-01. PDF (accessed 2017-02-13)*, 2012.

[33] C. D. e. Manning, *Introduction to information retrieval*, vol. 1. Cambridge university press Cambridge, 2008.

[34] A. Dorri, S. S. Kanhere, and R. Jurdak, "Blockchain in internet of things: challenges and solutions," *arXiv preprint arXiv:1608.05187*, 2016.

[35] R. Henry, F. Olumofin, and I. Goldberg, "Practical pir for electronic commerce," pp. 677–690, 2011.

[36] A. Kate, G. M. Zaverucha, and I. Goldberg, "Constant-size commitments to polynomials and their applications," in *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 177–194, Springer, 2010.

[37] I. Damgård, "Commitment schemes and zero-knowledge protocols," in *School organized by the European Educational Forum*, pp. 63–86, Springer, 1998.

[38] S. Nath, "Madscope: Characterizing mobile in-app targeted ads," in *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, pp. 59–73, 2015.

[39] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," tech. rep., Manubot, 2019.

[40] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: A survey," *International Journal of Web and Grid Services*, vol. 14, no. 4, pp. 352–375, 2018.

[41] A. Dorri, S. S. Kanhere, and R. Jurdak, "Mof-bc: A memory optimized and flexible blockchain for large scale networks," *Future Generation Computer Systems*, vol. 92, pp. 357–373, 2019.

[42] A. Bogdanov, M. Knežević, G. Leander, D. Toz, K. Varıcı, and I. Verbauwhede, "Spongent: A lightweight hash function," in *International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 312–325, Springer, 2011.

[43] S. Englehardt and A. Narayanan, "Online tracking: A 1-million-site measurement and analysis," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pp. 1388–1401, 2016.

[44] A. Frik, A. Haviland, and A. Acquisti, "The impact of ad-blockers on product search and purchase behavior: A lab experiment," in *29th {USENIX} Security Symposium ({USENIX} Security 20)*, 2020.

[45] W. Meng, R. Ding, S. P. Chung, S. Han, and W. Lee, "The price of free: Privacy leakage in personalized mobile in-apps ads.," in *NDSS*, 2016.

[46] A. Shuba and A. Markopoulou, "Nomoats: Towards automatic detection of mobile tracking," *Proceedings on Privacy Enhancing Technologies*, vol. 2, pp. 45–66, 2020.

[47] U. Iqbal, P. Snyder, S. Zhu, B. Livshits, Z. Qian, and Z. Shafiq, "Adgraph: A graph-based approach to ad and tracker blocking," in *Proc. of IEEE Symposium on Security and Privacy*, 2020.

[48] G. Merzdovnik, M. Huber, D. Buhov, N. Nikiforakis, S. Neuner, M. Schmiedecker, and E. Weippl, "Block me if you can: A large-scale study of tracker-blocking tools," in *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 319–333, IEEE, 2017.

[49] A. Das, G. Acar, N. Borisov, and A. Pradeep, "The web's sixth sense: A study of scripts accessing smartphone sensors," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1515–1532, 2018.

[50] X. Liu, J. Liu, S. Zhu, W. Wang, and X. Zhang, "Privacy risk analysis and mitigation of analytics libraries in the android ecosystem," *IEEE Transactions on Mobile Computing*, 2019.

[51] M. C. Grace, W. Zhou, X. Jiang, and A.-R. Sadeghi, "Unsafe exposure analysis of mobile in-app advertisements," in *Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks*, pp. 101–112, 2012.

[52] T. Book and D. S. Wallach, "A case of collusion: A study of the interface between ad libraries and their apps," in *Proceedings of the Third ACM workshop on Security and privacy in smartphones & mobile devices*, pp. 79–86, 2013.

[53] T. Book and D. S. Wallach, "An empirical study of mobile ad targeting," *arXiv preprint arXiv:1502.06577*, 2015.

[54] S. Demetriou, W. Merrill, W. Yang, A. Zhang, and C. A. Gunter, "Free for all! assessing user data exposure to advertising libraries on android.," in *NDSS*, 2016.

[55] V. F. Taylor, A. R. Beresford, and I. Martinovic, "Intra-library collusion: A potential privacy nightmare on smartphones," *arXiv preprint arXiv:1708.03520*, 2017.

[56] S. Seneviratne, H. Kolamunna, and A. Seneviratne, "A measurement study of tracking in paid mobile applications," in *Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, pp. 1–6, 2015.

[57] N. Vallina-Rodriguez, S. Sundaresan, A. Razaghpanah, R. Nithyanand, M. Allman, C. Kreibich, and P. Gill, "Tracking the trackers: Towards understanding the mobile advertising and tracking ecosystem," *arXiv preprint arXiv:1609.07190*, 2016.

[58] R. Binns, J. Zhao, M. V. Kleek, and N. Shadbolt, "Measuring third-party tracker power across web and mobile," *ACM Transactions on Internet Technology (TOIT)*, vol. 18, no. 4, pp. 1–22, 2018.

[59] R. Binns, U. Lyngs, M. Van Kleek, J. Zhao, T. Libert, and N. Shadbolt, "Third party tracking in the mobile ecosystem," in *Proceedings of the 10th ACM Conference on Web Science*, pp. 23–31, 2018.

[60] F. Roesner, T. Kohno, and D. Wetherall, "Detecting and defending against third-party tracking on the web," in *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, pp. 12–12, USENIX Association, 2012.

[61] A. Lerner, A. K. Simpson, T. Kohno, and F. Roesner, "Internet jones and the raiders of the lost trackers: An archaeological study of web tracking from 1996 to 2016," in *25th {USENIX} Security Symposium ({USENIX} Security 16)*, 2016.

[62] X. Pan, Y. Cao, and Y. Chen, "I do not know what you visited last summer: Protecting users from third-party web tracking with trackingfree browser," in *Proceedings of the 2015 Annual Network and Distributed System Security Symposium (NDSS), San Diego, CA*, 2015.

[63] M. Fredrikson and B. Livshits, "Repriv: Re-imagining content personalization and in-browser privacy," in *Security and Privacy (SP), 2011 IEEE Symposium on*, pp. 131–146, IEEE, 2011.

[64] H. Haddadi, P. Hui, and I. Brown, "Mobiad: private and scalable mobile advertising," in *Proceedings of the fifth ACM international workshop on Mobility in the evolving internet architecture*, pp. 33–38, ACM, 2010.

[65] R. Chen, I. E. Akkus, and P. Francis, "Splitx: High-performance private analytics," *SIGCOMM Comput. Commun. Rev.*, vol. 43, pp. 315–326, Aug. 2013.

[66] N. Mor, O. Riva, S. Nath, and J. Kubiatowicz, "Bloom cookies: Web search personalization without user tracking.," in *NDSS*, 2015.

[67] G. Zyskind, O. Nathan, *et al.*, "Decentralizing privacy: Using blockchain to protect personal data," in *2015 IEEE Security and Privacy Workshops*, pp. 180–184, IEEE, 2015.

Imdad Ullah has received his Ph.D. in Computer Science and Engineering from The University of New South Wales (UNSW) Sydney, Australia. He is currently an assistant professor with the College of Computer Engineering and Sciences, PSAU, Saudi Arabia. He has served in various positions of Researcher at UNSW, Research scholar at National ICT Australia (NICTA)/Data61 CSIRO Australia, NUST Islamabad Pakistan and SEEMOO TU Darmstadt Germany, and Research Collaborator at SLAC National Accelerator Laboratory Stanford University USA. He has research and development experience in privacy preserving systems including private advertising and crypto-based billing systems. His primary research interest include privacy enhancing technologies; he also has interest in Internet of Things, Blockchain, network modeling and design, network measurements, and trusted networking.

Salil S. Kanhere (Senior Member, IEEE) received the M.S. and Ph.D. degrees from Drexel University, Philadelphia. He is currently a Professor of Computer Science and Engineering with UNSW Sydney, Australia. His research interests include the Internet of Things, cyberphysical systems, blockchain, pervasive computing, cybersecurity, and applied machine learning. He is a Senior Member of the ACM, an Humboldt Research Fellow, and an ACM Distinguished Speaker. He serves as the Editor in Chief of the Ad Hoc Networks journal and as an Associate Editor of the IEEE Transactions On Network and Service Management, Computer Communications, and Pervasive andMobile Computing. He has served on the organising committee of several IEEE/ACM international conferences. He has co-authored a book titled *Blockchain for Cyberphysical Systems*.

Roksana Boreli has received her Ph.D in Communications from University of Technology, Sydney, Australia. She has over 20 years of experience in communications and networking research and in engineering development, in large telecommunications companies (Telstra Australia, Xantic, NL) and research organisations. Roksana has served in various positions of Engineering manager, Technology strategist, Research leader of the Privacy area of Networks research group in National ICT Australia (NICTA)/CSIRO Data61 and CTO in a NICTA spinoff 7-ip. Her primary research focus is on the privacy enhancing technologies; she also maintains an interest in mobile and wireless communications.