AnonAD: Privacy-aware Micro-targeted Mobile Advertisements without Proxies

Christopher Buckley*, Parth H. Pathak*, Aveek K. Das*, Chen-Nee Chuah†, Prasant Mohapatra*
*Computer Science Department, †Electrical and Computer Engineering Department,
University of California, Davis, CA, USA.
Email: {cmbuckley, phpathak, akdas, chuah, pmohapatra}@ucdavis.edu

***Abstract*—Mobile advertisements have become the dominant source of revenue for mobile application developers, advertisers and brokers. Using novel sensing techniques and the advanced sensors of mobile devices, it has become feasible to determine a user's fine-grained context such as her location, activity, and interests. This information can be used by the advertisement (ad) brokers to provide more relevant ads to the user based on her context. However, this has led to serious privacy risks, since a user can be tracked by the broker or an adversary based on her context. In this paper, we present AnonAd, an ad delivery scheme that allows users to protect their privacy when receiving micro-targeted ads from the broker. AnonAd utilizes the encryption of the user's context based on a split-secret scheme that guarantees that the broker can decrypt the context only when there exists k other users in the same context. This way, a user's privacy is protected with $k-$anonymity during the context report. We show that the split-secret scheme integrates seamlessly with existing homomorphic encryption-based schemes that can provide differential privacy for ad click reports. We implement AnonAd on Android smartphones and evaluate it with real users as well as simulated users that follow real mobility traces. Our results show that AnonAd achieves a balance between user's privacy and relevancy of advertisements without the requirement of any additional proxy servers.**

I. Introduction

As mobile devices become more effective at determining user context new opportunities are rising to provide micro-targeted advertisements. Various sensors (such as GPS, accerelometer, light sensor, etc.) on a mobile device now allow us to infer a user's context (e.g. location, activity, interests etc.) with much higher accuracy. More personalized ads can increase the chances of a user clicking on the advertisement which in turn can increase the revenue for the advertisers. In today's ad delivery system, brokers such as Yahoo and Google aggregate the ads from different advertisers. They also profile the users to understand their interest and context, and (ideally) serve them with an ad that is likely to be the most useful to the user. However, over the years, the brokers have become increasingly aggressive in their profiling, which has led to serious privacy concerns from users.

The privacy issue becomes even more severe with micro-targeted ads. In such a case, the broker has access to the user's micro-level context. Unlike most other advertising systems currently in use, which access broad context information (e.g. city-level location), micro-targeted ads are based on a user's micro-level context - their current activity, precise location, past interest profile, and so on. For example, a user's device can determine that the user is a "22 year female" who is "walking on 2nd street in San Francisco" and is "interested in Italian food." With this level of detail, it is possible for the broker, or an adversary, to determine the user's true identity. The infringement of a user's true identity is a much more severe problem than the infringement of her online identity (such as her current IP address). In this paper, we design an anonymity scheme that can protect a user's true identity while still allowing them to receive micro-targeted ads from an ad broker.

In our work, we achieve user anonymity by obfuscating the user's identity among k users - a concept known as *k-anonymity* [1]. Most of the current schemes that can achieve anonymity utilize third-party servers or proxies to aggregate and hide the user identities before forwarding their context to the broker [2], [3]. This way, the proxy acts as an intermediate point between the users and the broker. However, we hold that the use of these proxy servers is undesirable due to two main reasons: first, the assumption that such proxy servers are completely or even partially trusted is unrealistic in the real world. Offloading crucial security and privacy tasks to the proxy in fact makes them more vulnerable and does not necessarily improve user privacy. Second, considering the current business model of ad delivery systems, there is no economical incentive for brokers or advertisers to deploy such proxies, which makes users even more vulnerable to privacy risks. In this work, we propose a privacy-aware ad delivery system, *AnonAd* (Anonymized Advertisements), that does not require any third party servers or proxies in order to guarantee user privacy. We design an anonymization scheme based on split-secret and homomorphic encryption which can guarantee that user context reports and click reports are anonymized with bounded privacy guarantees.

AnonAd provides two levels of privacy guarantees. It protects the user's true identity by ensuring that broker can only know the user's true context when there are at least $k-1$ other users in the same context. We use split-secret key-based encryption where users only report part of the key that is derived from their context. When the broker has k such distinct key parts, it can decrypt the context. This ensures that the users are indistinguishable from $k-1$ other users in the same context. Once the user has received a micro-targeted ad from the broker, AnonAd protects their click reports using homomorphic encryption. Here, AnonAd ensures

978-1-4799-9964-4/15/$31.00 ©2015 IEEE

that the broker can only know the *sum of clicks* for all users while hiding the individual click reports of users. AnonAd protects the privacy of the user's context and click reports without requiring any additional proxy servers.

The contributions of this work can be summarized as follows:

(1) We present a privacy-aware micro-targeted ad delivery framework (AnonAd) that utilizes split-secrets to ensure user privacy. In AnonAd, users send their encrypted context to the broker, but the broker cannot decrypt the context unless there are $k-1$ other users in the same context. By obfuscating the user's identity among k users, AnonAd prevents the tracking of the user while still allowing the broker to provide micro-targeted ads to the users.

(2) We show that the split-secret scheme of context reporting integrates seamlessly with homomorphic encryption-based privacy protection schemes for click reporting. AnonAd does not require any additional proxies or communication among the users to ensure anonymity in either the context or click reporting phase.

(3) We implement AnonAd as an Android application and evaluate resource usage and ad relevancy with real users. We also evaluate AnonAd with simulated users that follow real-world mobility traces collected from a campus area network. Our evaluation shows that AnonAd achieves a balance between privacy and ad relevancy with some overhead of additional message exchange between the users and the broker.

The rest of the paper is organized as follows. We discuss the related work in Section II. The details of our system model and preliminaries of the ad delivery system are provided in Section III. Section IV provides an overview of the requirements and challenges in design of AnonAd. The AnonAd scheme is described in Section V with its security analysis in Section VI. The performance evaluation of AnonAd is provided in Section VII. We conclude our paper in Section VIII.

II. Related Work

Over the last few years, there have been a few research works related to mobile ad delivery systems. [2] presents the problem of personalized ad delivery as a three-way optimization problem. Like [2], our work also uses a tree structure for contexts. The main limitation of this work is the requirement for two separate servers to be used in the system. As previously mentioned, one of our goals is to avoid using any proxies or other "extra" servers if possible. The problem of personalized ad delivery has also been addressed in [3], but similarly to [2] they only solve it by adding an "honest but curious dealer" to their system. This adds another point of failure to the system, and allowing for this sort of dealer is an unrealistic assumption for real-world applications and their ecosystem.

Many other recent research works have used $k-$anonymity as a privacy guarantee for a variety of services such as location services [4], [5], health data [6], etc.

The application of homomorphic encryption with noise addition to provide differential privacy has been discussed in [7] and [8]. Such techniques allow sharing of personal data to untrusted servers such that the servers can only know a certain statistic (e.g. sum) of the responses from the users without knowing the actual response of individual users. These techniques were extended in [9] to address user churn (dynamic joining and leaving of users) and in [10] to enable non-tracking web analytics. In our click-report phase, we borrow these techniques and show that it can seamlessly integrate with our context report phase. Finally, we use the advertising architecture described in [2], [11] as the architecture for our advertising model.

III. System Model

AnonAd is designed to address one fundamental shortcoming of all current privacy-aware ad systems [2], [3], [10], which is that they all rely on third-party servers or proxies in order to guarantee user privacy. Such systems offload the tasks of ensuring user privacy from the broker to anonymizing proxies, which merely shifts the vulnerable end points in the systems without truly solving the privacy challenges. Additionally, AnonAd is designed to fit in the current real-world ad delivery model which has already proven to be economically viable. We believe that providing privacy guarantees without requiring any modification to the current ad delivery system is one of the most salient features of AnonAd. It contains the following four entities which also constitute most real-world ad delivery systems:

User: A user represents a mobile client who will receive a micro-targeted ad based on their context. We assume that users want to maximize ad relevance (i.e. receive ads that are relevant to their interests and location), but also want to maintain a certain degree of anonymity. This tension between relevance and anonymity is a recurring theme in our work.

Broker: Ad brokers are responsible for delivering advertisements from advertisers to the users. Example of major ad brokers are Google and Yahoo, among many others. The broker's goal is to maximize revenue, which is often - but not always - correlated with relevance [2]. We consider the revenue/relevance disparity to be beyond the scope of this paper, and thus assume that maximizing ad relevance will also maximize broker revenue. We also assume that the broker is untrusted - that is, the broker will attempt to learn as much as possible about the users and attempt to exploit this knowledge. One of our goals is to allow the broker to maximize their revenue without sacrificing user privacy.

Advertiser: Advertisers are corporations, businesses, or other organizations that want to reach out to a specific demographic of users. In most ad systems, advertisers pay the ad broker for displaying their ad, as well as for each click on an ad.

Key Distribution Agency (KDA): The KDA is responsible for distributing secrets which are used in our algorithms for context and click reporting. We describe the secrets in more detail in Section V. We assume that the KDA is untrusted but honest - users will not trust them with any of their information (in fact, the KDA doesn't need any user information in order to perform its job), but will abide by the rules of the system,

distributing the secrets appropriately. In today's ad delivery systems, this job is best suited for a certificate authority (CA). Note that the KDA can be eliminated when users can generate secrets in a distributed manner. Because of this, and the fact that the KDA isn't ever exposed to any user data, we don't count it as a proxy/extra server. However, for simplicity, we'll assume that there exists a dedicated KDA for key distribution.

The operations of AnonAd can be divided into two main phases:

(1) Ad Distribution Phase: This phase involves users determining their context, encrypting it, and uploading it to the broker, who may only decrypt this context if there are at least $k-1$ other users who are also in that same context. If the broker successfully decrypts a context, an appropriate ad will be distributed to the users in that context; otherwise, the users will generalize their context and retry. This phase is considered over once the user has received an ad.

(2) Click Report Phase: Once a user has received and viewed an ad, they may decide to click on it if it is relevant to their interests. This phase involves ensuring that this is done anonymously, since a user's click patterns could potentially violate their privacy. For this part, we borrow a solution proposed by [8], [9], which use homomorphic encryption along with noise addition to ensure that the user's identity is protected under differential privacy guarantees.

Next we outline the privacy requirements of both the phases, and the threat model.

IV. Requirements & Challenges

In this section, we look at the requirements of our model and also look at how the different entities involved can attempt to compromise user privacy.

A. Privacy Goals

In order to provide anonymity in the *ad distribution phase*, we use the well-known concept of k-*anonymity* . While k-*anonymity* has been shown to have some weaknesses [12], we hold that it is actually sufficient for our needs. It would be possible to extend our system to use more advanced techniques such as ℓ-*diversity* [12], but many of the attacks on k-*anonymity* require the use of outside knowledge which will mostly be unavailable to potential attackers using our system.

For the *click report phase*, we use *differential privacy* techniques. Differential privacy is a preferred choice here, as we are aggregating information across a user base (in this case, the click reports of the users) and want to ensure that we get reasonable, strongly-bounded results without being able to identify who actually clicked on the ad.

B. Auxiliary Goals

While anonymity is the main focus of our system, there are several auxiliary goals that we would like to fulfill as well.

No Proxies: As mentioned before, one of the other main goals of the AnonAd system is to avoid the use of proxies or any additional servers. In particular, this includes *any* extra entity outside of the four "core" entities mentioned above: user, broker, advertiser, and KDA.

Using proxies, even proxies that are only "semi-trusted," as in [3], allows for aggregation between the users and the broker/advertiser. Thus, all the users can communicate with the proxy, eliminating any need for the users to communicate among themselves thwarting any potential user-user collusion attacks. The users never need to deal directly with the broker, protecting their identities and also preventing user-broker collusion attacks. However, we claim that using this sort of proxy is unrealistic in the real world, as it would have to be run by some entity, whether it be an individual or a corporation, and that entity would then be in a position to violate user privacy. Some works (e.g. [3]) make use of a "semi-trusted" or "honest but curious" proxy, but even these proxies still add another point of failure and vulnerability in the system, thus weakening the scheme overall.

AnonAd was designed to provide user privacy without requiring any proxies. Although our scheme does require the KDA, we show that this could actually just be a certificate authority (henceforth a "CA"). CAs are already deployed and widely used and available under the current Internet model. Additionally, the CA never actually learns anything about the user, aside from some ephemeral information such as their IP address. Finally, we also show that it's even possible to eliminate the KDA entirely, so long as their exists a scheme by which all users can derive the key independently (one possibility uses hashed timestamps, using the resultant hash as the key).

Communication Cost: A consequence of not utilizing proxies is that our system will likely require more message transmissions. With this in mind, we want to ensure that the extra cost incurred is not too high, and that our system will scale well with increasingly large user bases. Naturally, this is going to have subtle interactions with other parts of our system. It is also necessary to ensure that there is no need for direct communication between the users. This precludes the use of any distributed solutions where users collaboratively derive the necessary security parameters (secrets, etc.). This also eliminates the possibility of user collusion attacks wherein a certain subset of users can coordinate to leak the privacy of other users in the system.

Since message transmissions tend to be strongly correlated with other forms of resource usage on smartphones (e.g. more message transmissions cause the battery to drain faster), we only consider communication costs in this work, and assume that the same results apply to other smartphone resources as well (battery, CPU usage, etc.).

C. Security & Threat Model

In addition to the above requirements, our system must also be secure. Here, we revisit the four entities from Section III in order to show how they may attempt to attack user privacy.

User: A malicious user attempts to violate the privacy of other users, generally by colluding with the broker, or

potentially the advertiser. This can be accomplished by the release of a secret key to the broker.

Broker: A malicious broker attempts to collect as much information about the users as possible, in order to exploit this information. We assume that any information available to the users is also available to the broker, including public encryption keys and algorithms. One avenue of attack for the broker is to attempt an exhaustive search attack across all possible contexts. They can also collude with users, as mentioned above.

Advertiser: For the most part, we do not consider the advertiser as a threat, since they are only connected to the users through the broker. However, they may be able to collude with users in order to learn about user preferences, which in turn would give them an advantage in negotiating with the broker (since they would know which ads are more likely to be popular). Nonetheless, we consider this threat to be beyond the scope of this work.

KDA: We assume that the KDA is untrusted but honest. This means that the users will never trust them with any personal data. A malicious KDA can give out mismatched public keys during the ad distribution phase, but this attack serves no purpose other than to disrupt the system. Alternatively, they could provide incorrect integers in the click report phase that do not sum to zero (which is one of the requirements of our algorithm in that phase). This will not have any effect on user privacy but does make it more difficult for the broker to function properly in that phase.

V. AnonAd Scheme

As mentioned previously, the AnonAd scheme features two main phases: *Ad Distribution Phase* and *Click Report Phase*.

A. Ad Distribution Phase

In the *ad distribution phase*, each user reports their context and receives a corresponding ad *if and only if* there are enough other users in the same context to guarantee that their privacy will be preserved. We consider privacy to be preserved if the context meets the requirements of k-anonymity - i.e., there are at least $k-1$ other users in the same context. If there aren't enough users in the same context, the context is generalized and the process is repeated again after a delay.

(1) At the start of the phase, every user will query the KDA in order to receive a public hash key K^+.[1] Note that this key is *public* - that is, we assume that the broker knows what this key is. As we will show below, this knowledge does not help the broker violate user privacy. It is *critical* to our scheme that this hash key is changed periodically, since a malicious broker could otherwise store past contexts and break the encryption (see *Step 3* and *Step 4*, below).

(2) Next, each user u should determine their context, c_u. The context is a set of user attributes that can be used by the broker to serve a micro-targeted ad. In our system, a context is a vector of attributes, consisting of an interest and a location. We concatenate multiple contexts when they are being reported simultaneously. AnonAd can handle a wide variety of context representations - the only requirements are that they are unique, can be encrypted/hashed, and are "generalizable." A "generalizable" context is one that can be generalized to a broader, less specific context. This generalized context should be a superset of the original one, and should include other related contexts. As an example, imagine a user's context is {Location = 1st Street, Davis, CA & Interest = Baseball}. This context can be generalized to either {Location = Davis, CA & Interest = Baseball} or {Location = 1st Street, Davis, CA & Interest = Sports}, depending on the system implementation.

(3) Now, each user will hash their context using a cryptographic hash function H and key K^+. This results in $H_{K^+}(c_u)$, which will henceforth be referred to as K^-, or the "secret key." If two users are in the same context, they should both end up with the same key K^- - i.e., $H_{K^+}(c_i) = H_{K^+}(c_j)$ iff $c_i = c_j$.

(4) Each user then encrypts their context c_u using an encryption function $\mathcal{E}$, with the secret key K^- as the key. We denote the result to be $\mathcal{E}_{K^-}(c_u)$, or $\mathcal{E}(c_u)$ for short. As with the hash in *Step 3*, we expect that $\mathcal{E}(c_i) = \mathcal{E}(c_j)$ iff $c_i = c_j$, since $[c_i = c_j] \iff [H_{K^+}(c_i) = H_{K^+}(c_j)]$ with very high probability.

(5) Next, each user will generate a token t_u from the bits of the secret key K^-. The key should be broken down into k uniformly distributed chunks, each approximately of size $sizeof(K^-)/k$, where $sizeof(K^-)$ is the number of bits in K^-, and k is the constant for k-anonymity. Each user should select one of these chunks with uniformly random probability to be their token. This token will be used in a *secret sharing* scheme later in the phase. Tokens also include metadata indicating which part of the secret key they are from - i.e., the range of bits that the token represents. This will help the broker re-assemble the key later.

(6) Now, each user will send the message $\{\mathcal{E}(c_u), t_u\}$ to the broker. The broker will accumulate these messages, maintaining a mapping from the encrypted contexts to the tokens. Each time they receive a token, they should combine it with the other tokens that they've already accumulated for that context. Once they have collected every token, they can assemble the secret key K^- and decrypt the context $\mathcal{E}(c_u)$.

This step has one critical property for our system: the broker will require *at least* k tokens in order to assemble the key. This follows naturally from the fact that there are k different chunks using the pigeonhole principle. This guarantees that by the time the broker is able to decrypt the context, there are k users in the same context, which means that the users will be protected under k-anonymity. Determining how many users are necessary to decrypt a context is essentially a version of the *coupon collector's problem*.

(7) If, after an implementation-defined timeout, the broker is unable to decrypt a given context, the users should generalize their contexts and the process should be repeated from scratch.

[1] As mentioned previously, the KDA can actually be eliminated here - the only requirement is that all users end up with the same key. One possibility is to hash timestamps, using the hash as the key.

Eventually, the users will either receive an ad, or generalize to a "root" level, indicating that their context is too specific or unique to be used for generating an ad.

Note that the more a context is generalized, the less specific - and therefore less relevant - the returned ads will be. As suggested by [2], there is a trade-off between privacy, relevancy, and efficiency - more privacy will generally lead to less ad relevance and/or efficiency.

B. Click Report Phase

Once the broker distributes an ad to the k users in the same context, it is necessary for the broker to evaluate the relevancy of the ad by knowing whether the users clicked on the ad or not. This requires a click report to be sent from the user to the broker. However, in order to protect the user's privacy, it is necessary that the broker only receives the total sum of clicks as opposed to individual clicks of the users. [8] presented a solution that uses homomorphic encryption along with additional noise to protect the user's identity while allowing the broker to retrieve the (noisy) sum of clicks. Let $x_i \in \{0, 1\}$ be the click report from user i where $x_i = 1$ if the user clicked on the ad and $x_i = 0$ otherwise. In order to ensure that broker can only retrieve the sum of $\{x_1, x_2, \cdots x_k\}$, the KDA assigns $k + 1$ randoms shares of 0 to k users $\{r_1, r_2, \cdots, r_k\}$ and the broker (r_0) such that $\Sigma_{i=0}^{k} r_i = 0$. In order to provide differential privacy guarantee [8], each user first adds a noise n_i and the random secret r_i to her x_i and finds $x_i^* = x_i + n_i + r_i$. The response x_i^* is then sent to the broker. The broker finds the noisy sum by calculating the sum $r_0 + \Sigma_{i=1}^{k} x_i^*$. Due to $\Sigma_{i=0}^{k} r_i = 0$, the broker can retrieve the $\Sigma_{i=1}^{k} x_i + n_i$. Because the scheme is shown to provide differential privacy guarantee [8], [9], we use this scheme in the click report phase of AnonAd. Note that the $k+1$ random share of 0 can be provided by the KDA to users and the broker after an ad is delivered to the users.

VI. Security Analysis

Here we examine and formalize some of the security properties of our system. The security properties of the click-report phase have been analyzed in [8], [9], so we primarily focus on our context-report phase here.

(1) The broker cannot gain any advantage by using the public key K^+ to derive a user's secret key K^-: In order to derive a user's secret key K^- by using the public key K^+, the broker would need to first know the user's context c_u, then hash it using the cryptographic hash function H with key K^+. However, there is no advantage in the broker doing this, because in order to do this they would have already had the context c_u, defeating the entire purpose of obtaining K^- in the first place. Therefore, the broker gains no advantage by deriving K^- in this manner.

(2) The broker cannot easily derive the secret key K^- through cryptanalysis: Deriving K^- using any sort of cryptography-based attack essentially amounts to the problem of breaking the cryptographic hash function H. Assuming that H is a "good" (i.e., collision-intractible) hash function, this is prohibitively difficult in this context.

(3) The system is resistant (but *not* immune) to client-broker collusion attacks: Users have no incentive to participate in a client-broker collusion attack (e.g., revealing their secret key K^-), as they are only violating their own privacy in doing so. The broker could set up a "rogue" client in an attempt to obtain K^-, or alternatively traverse the context tree and hash each context manually. However, this attack has two flaws:

(a) The cost of this attack is proportionate to the number of contexts in the context tree. If more contexts are added, the broker has to perform more computations in order to break them.

(b) The public key K^+ is constantly changing, thus making previously-derived secret keys invalid after a timeout.

This attack can also be mitigated by using a scheme inspired by BitCoin/Hashcash [13], [14]: instead of using the hash $H_{K^+}(c_u)$ as the secret key K^-, a chain of hashes could be created, with the result of the final hash in the chain being used as K^-. In order to derive the secret key, the entire chain would have to be traversed. This does require the clients to perform additional computation, but they only have to do it for a small subset of the contexts - the broker would have to do it for the entire tree, which would become prohibitively expensive.

(4) Users are protected under k-*anonymity*: In order to assemble the secret key K^-, the broker must have *at least* k tokens from k different users - as mentioned previously, this follows naturally from the pigeonhole principle.

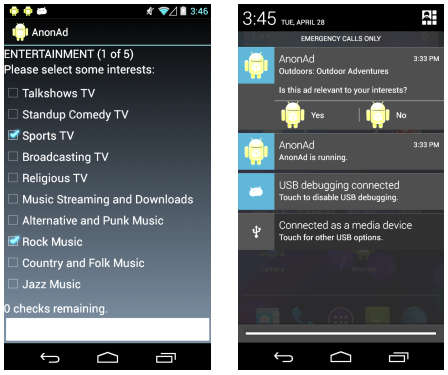
It should be noted that the closer the broker is to accumulating the full key, the easier a brute-force attack becomes - for example, if there are only 10 bits of entropy left, the broker only has to guess from $2^{10} = 1,024$ combinations in order to derive K^- and decrypt the context. One solution to this problem is to make the key chunks smaller, and *force* the broker to brute-force the remainder of the key. For example, with a 128-bit key and $k = 4$, 24-bit chunks could be used. Once the broker has accumulated the four 24-bit chunks (a total of 96 bits), they would need to guess the remaining 32-bits. This does create additional work for the broker, but it also means the individual chunks will not affect the entropy as much, making the system more resistant to brute-force attacks. As with other parts of our system, there is a trade-off between privacy and efficiency here.

VII. Evaluation

In this section, we describe the details of the AnonAd app implementation, experiment setup and performance evaluation. Since the ad distribution phase is our central contribution in this work, we primarily focus on that phase in our evaluation.

A. Implementation

We implemented the AnonAd scheme as a stand-alone application on Android smartphones (Android version 4.1+). Figs. 1a and 1b show the app implementation. As shown in



(a) Interest selection on AnonAd app

(b) Ad received by the user in the AnonAd app

Fig. 1: AnonAd Android App Implementation

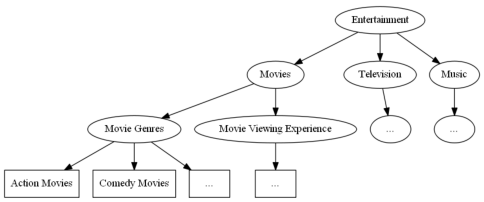


Fig. 2: A sampling of nodes from our entertainment context tree. The boxed nodes are leaves. Each round, the app starts from the leaves and generalizes upwards until either an ad is received or no further generalization is possible.

Fig. 1a, when the app is run for the first time, it prompts the user to select their interests for five different interest categories. The categories are "entertainment," "food & drink," "healthcare," "shopping," and "sports." These interest categories are arranged in a tree format, with the "higher" elements of the tree being generalizations of their children. For example, "Italian Restaurant" is a type of "European Restaurant," which can be generalized to "Ethnic Restaurants," then just "Restaurants," and finally "Food & Drink" - the "root" element of the tree. In order to ensure there is sufficient ad variety for our experiments, the user is required to select at least one interest from each category, and up to three total *per category*. The user is also only allowed to select from the leaf elements of the tree (i.e., in the above example, they cannot select "Ethnic Restaurants" - only "Italian Restaurants"). These interests will be combined with location information in order to obtain a context for each user.

The AnonAd app runs in the background and reports its context periodically. In our implementation, we configured the app to report user's context every half an hour (i.e. 3PM, 3:30PM and so on) for a "round." The reason we choose to synchronize the context reporting is because of the limited number of users participating in our experiments. If AnonAd is deployed at a large scale (hundreds to thousands of users), this syncronization would not be necessary. Each ad round has an associated interest category, which we determined based on what we expected the user to be interested in at that time. For example, from 5:30-7pm, the app will use the "food & drink" category, based on the assumption that the user is most likely interested in dinner restaurants at that time.

At the start of each ad round, the app determines the interest category associated with the timeslot, and then retrieves *all* of the interests the user selected for that category. The app then determines the current user location using both GPS and network data. Like the interest categories, the user locations are also represented by a tree - so the bottom level elements represent fine-grained locations (e.g., street/block level), and the higher level elements are less precise (e.g., district level).[2] Initially, the most precise location available is used.

Once the user has determined their location, this information is combined with their interests to form the contexts. The contexts are then encrypted and uploaded to the "broker" (implemented on a Linux server). Since the public KDA key is actually only necessary to prevent what we refer to as "broker memory" attacks (in which the broker remembers past context encryptions and maps them back to a previously decrypted context), we chose to use a constant public key for *Step 1* in order to simplify the app design. The encryption algorithm we use is AES with the CBC mode of operation and a 128-bit key, and the hashing algorithm we use is HMAC-SHA1. The secret keys used for encryption are generated by running the context through multiple rounds of hashing; in each round, the first 32-bits of the resultant 160-bit hash are used as a piece of key, and the remaining 128-bits are used as the key for the next round of hashing. The k-value is retrieved from our server and can be dynamically altered during the experiments.

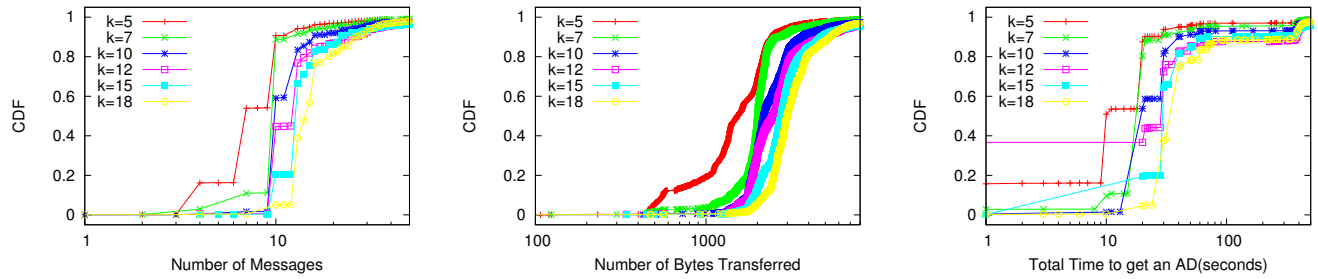
If the context is successfully decrypted (there are at least $k-1$ other users in the same context), the broker will send out an ad to the k users. The ad appears as a notification which the user can respond to based on whether the received ad is relevant to their interests or not. If the context cannot be decrypted, the app generalizes the context and tries to get an ad using the generalized context. If no further generalization is possible (i.e., the app has generalized to the root node), the entire process is restarted from scratch after a timeout (5 minutes in our case).

The ads deployed by the server once the context has been decrypted are based on the Google Places API [15]. The API is queried by the server using the context and user location as reported by the user. The API responds with businesses that are related to the search string and are within a certain predefined distance of the location. This business information is pushed as an ad to the users.

B. Experiment Setup

Since only a limited number of users can evaluate the AnonAd scheme using the Android app, we simulated additional clients to evaluate larger values of k. For the experiments, we installed the app on seven different Android devices, and one-hundred additional users were simulated. The user interests of the simulated users were generated

[2]For the purposes of our app and experiments, only locations within Davis, California are used.



(a) (b) (c)

Fig. 3: (a) Number of messages sent from client before it receives an ad, (b) Upstream bytes from client to server before an ad is received and (c) Delay between the first context reporting and an ad being received

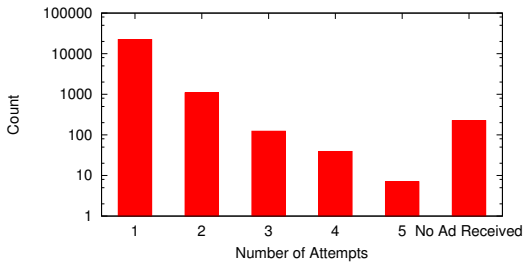


Fig. 4: Number of attempts before ad delivery

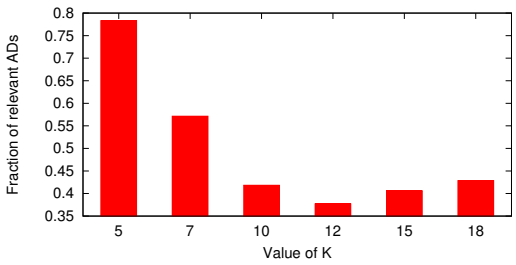


Fig. 5: Relevance of ads as compared to k

uniformly at random by the simulator. We use real mobility traces collected from the university campus network during our previous study [16] for the locations of the simulated users. The mobility traces provide the user's location based on which WiFi access points the user connects to on the university campus network. We have anonymized the mobility traces to remove any personally identifiable information from the traces [17]. We apply direct scaling on the mobility traces to emulate city-level mobility. Since the locations were generated from real mobility data, the simulated users would cluster at certain locations, which accurately reflects reality (e.g., people tend to cluster at supermarkets, cafés, etc.).

The simulated users allow us to evaluate large values of k, and the impact of different k-values on resource usage and delay. However, for evaluating ad relevancy, we only rely on the AnonAd app users, as the simulated users cannot provide any meaningful relevancy response.

C. Numerical Evaluation

In this section, we present some of the results of the experiments performed using our implementation, using the set-up described above. Specifically, we will evaluate how the anonymity requirement impacts the performance of ad delivery and resource utilization on client devices.

Number of messages/bytes: We first evaluate the number of messages that a client device needs to send the broker before it can receive an ad. Fig. 3a shows the cumulative distribution function of number of messages for different values of k. In our application protocol, a message is an encrypted context (or multiple contexts) sent from the client to the broker, or a reponse from the broker. However, the broker only sends messages in response to queries for the k-value, to send an ad, or in acknowledgement of something sent by the client. These messages are actually only a small fraction of the total messages sent in our system, so we focus on upstream client messages. Fig. 3b shows the number of bytes sent from the client to the broker before it received an ad. Note that we only included the application-layer messages and bytes and ignored the underlying TCP connection overhead (which is only a small, constant fraction of the overall traffic).

Figs. 3a and 3b show that the number of messages and bytes sent increases with an increase in the value of k. This shows the trade-off between the resource usage (communication cost) and the achievable privacy. Typically, user devices have to send more encrypted context messages in order to achieve stronger privacy guarantees. This is consistent with the privacy and relevance/efficiency trade-off detailed in [2]. A closer examination shows that the efficiency penalties are much higher in comparison with the relevance penalities, since at a higher value of k, communication cost increases much faster than the decrease in ad relevancy (see Fig. 5). However, we also see that even with a higher k-value, the efficiency impact is not as serious as one might think - regardless of the k-value, the majority of users (80+%) needed to send no more than 4,000 bytes of data in order to receive an ad. For comparison, a 240x200 pixel JPEG ad downloaded from the internet was 4.15 kilobytes, or approximately 4,150 bytes. For lower k-values (5 and 7, which we claim is still sufficient for maintaining user privacy), most users only needed to send around 2,000 or fewer bytes of data to receive an ad.

Delay: Another consideration for our system is the time it takes before an ad is received by the end user. Fig. 3c shows the cumulative distribution function of delay between the time when the first context report was sent and when an ad was received. We observe that it is relatively uncommon for it to take more than 100 seconds to receive an ad, regardless of the

k-value (over 80% of users received an ad within 100 seconds). For the low-to-mid k-values (5 to 10), most users would receive an ad in less than a minute. Clients who attempted to fetch an ad after the initial burst of requests would often receive a response within a few seconds, since most of the common contexts would have been decrypted by then. Note that the shape of the time graph - steep inclines followed by long stretches of flatness - comes from our system's timeout system: when an ad isn't received after the first attempt, the system will back off for a certain amount of time (5 minutes) and then make another attempt. The incline at 400 seconds represents the users who are receiving an ad after a retry, which typically would happen about 6 minutes after the round's start (accounting for an extra minute or so to communicate with the server), and the flat stretches are the periods where the users are waiting for the timeout.

Both the number of messages sent and the delay before an ad can be received are related to the number of attempts a client device has to perform in order to receive the ad. Fig. 4 shows the distribution of the number of tries it required for clients to receive an ad. Note that for a higher value of k, more attempts are expected given that more generalization is required to guarantee anonymity. We can observe from Fig. 4 that the majority of the time, an ad is received with a single attempt. Although relatively rare, there are some cases where an ad is not delivered to the user. Since this is necessary to guarantee user privacy, we suggest configuring the broker to deliver a generic ad (irrelevant to user context) in cases where the user's context cannot be decrypted within a given time period.

Ad relevance: Lastly, we take a look at the relevancy of the ads provided by the AnonAd broker while protecting user anonymity. Because the simulated clients cannot attest to an ad's relevance, we only rely on the AnonAd Android app clients for the evaluation. Fig. 5 shows the fraction of ads users found relevant for different values of k. As expected, a higher value of k means more generalization is required which in turn leads to less relevant ads. The same phenomenon is observed in Fig. 5. We also observe that the variation in relevance reduces significantly at higher values of k (e.g. $k \geq 10$), likely indicating a sharp drop in ad quality beyond that point. The decrease in relevance at $k = 12$ followed by the minor increase is believed to be an anomaly introduced by how the context is generalized in our implementation. Specifically, if the k-value is sufficiently high and the majority of users are in a relatively small subset of the possible contexts (which would be the case due to our simulator's use of real location traces), users outside of that subset would generally not receive an ad at all. On the other hand, users in that subset of contexts would receive a relevant ad, since there are so many other users clustered in that same context. Thus, instead of receiving less relevant ads, it would seem our users are either receiving no ad, or an ad that is still relevant to their interests. The problem can be solved by context pipelines, where multiple distinct contexts are reported simultaneously by the clients in order to allow parallel generalization that can benefit many clients in those contexts.

VIII. Conclusion

In this work, we presented AnonAd, a system for delivering micro-targeted ads without violating user privacy. We show that using split-secret based solution for context report, it is possible to eliminate proxy servers while still protecting user privacy. AnonAd provides $k-$anonymous privacy guarantees for the context report phase and differential privacy guarantee in the click report phase. We implemented and evaluated the AnonAd scheme as an Android app and with simulated users following real mobility traces collected from a university campus network. In our future work, we plan to explore user churn (dynamic joining and leaving of users) and its impact on AnonAd privacy. We will also explore user-user and user-broker collusion attacks to extend our AnonAd framework.

References

[1] L. Sweeney, "K-anonymity: A model for protecting privacy," *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, vol. 10, pp. 557–570, Oct. 2002.

[2] M. Hardt and S. Nath, "Privacy-aware personalization for mobile advertising," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, (New York, NY, USA), pp. 662–673, ACM, 2012.

[3] S. Guha, B. Cheng, and P. Francis, "Privad: Practical privacy in online advertising," in *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation*, NSDI'11, (Berkeley, CA, USA), pp. 13–13, USENIX Association, 2011.

[4] B. Gedik and L. Liu, "Protecting location privacy with personalized k-anonymity: Architecture and algorithms," *Mobile Computing, IEEE Transactions on*, vol. 7, pp. 1–18, Jan 2008.

[5] R. Shokri, C. Troncoso, C. Diaz, J. Freudiger, and J.-P. Hubaux, "Unraveling an old cloak: K-anonymity for location privacy," in *Proceedings of the 9th Annual ACM Workshop on Privacy in the Electronic Society*, WPES '10, (New York, NY, USA), pp. 115–118, ACM, 2010.

[6] K. El Emam, F. K. Dankar, R. Issa, E. Jonker, D. Amyot, E. Cogo, J.-P. Corriveau, M. Walker, S. Chowdhury, R. Vaillancourt, T. Roffey, and J. Bottomley, "A globally optimal k-anonymity method for the de-identification of health data," *Journal of the American Medical Informatics Association*, vol. 16, no. 5, pp. 670–682, 2009.

[7] E. Shi, T.-H. H. Chan, E. G. Rieffel, R. Chow, and D. Song, "Privacy-preserving aggregation of time-series data.," in *NDSS*, vol. 2, p. 3, 2011.

[8] E. Shi, T.-H. H. Chan, E. G. Rieffel, R. Chow, and D. Song, "Privacy-preserving aggregation of time-series data.," in *NDSS*, vol. 2, p. 3, 2011.

[9] T.-H. H. Chan, E. Shi, and D. Song, "Privacy-preserving stream aggregation with fault tolerance," in *Financial Cryptography and Data Security*, pp. 200–214, Springer, 2012.

[10] R. Chen, I. E. Akkus, and P. Francis, "Splitx: High-performance private analytics," in *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, SIGCOMM '13, (New York, NY, USA), pp. 315–326, ACM, 2013.

[11] S. Nath, "Madscope: Characterizing mobile in-app targeted ads," in *International Conference on Mobile Systems, Applications, and Services (MobiSys'15)*, ACM, May 2015.

[12] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, "L-diversity: Privacy beyond k-anonymity," *ACM Trans. Knowl. Discov. Data*, vol. 1, Mar. 2007.

[13] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2009.

[14] A. Back, "Hashcash - a denial of service counter-measure," 2002.

[15] "Google Places API." https://developers.google.com/places/.

[16] A. K. Das, P. H. Pathak, C.-N. Chuah, and P. Mohapatra, "Characterization of wireless multi-device users," in *IEEE SECON, 2015*.

[17] J. Fan, J. Xu, M. H. Ammar, and S. B. Moon, "Prefix-preserving ip address anonymization: Measurement-based security evaluation and a new cryptography-based scheme," *Comput. Netw.*, vol. 46, pp. 253–272, Oct. 2004.