DataBank: An Economics Based Privacy Preserving System for Distributing Relevant Advertising and Content

Rajan M. Lukose, Mark Lillibridge
Information Services and Process Innovation Laboratory
HP Laboratories Palo Alto
HPL-2006-95
June 26, 2006*

distributed systems, privacy, cryptography, economics, advertising, personalization, search

A critical feature of successful new advertising models has been their reliance on knowledge of various types of personal user data in order to make advertisements relevant and useful. This has raised many concerns about privacy and control over personal data. The preservation of privacy would appear to be in direct conflict with the successful new advertising models, which depend on knowledge of personal user data.

Here we present a system and its associated advertising model that shows this need not be the case. DataBank is a system for the delivery of relevant advertising (and content more generally) while preserving the privacy of user data as much as possible. The system relies on an economic pricing mechanism, similar in spirit to economic approaches to spam, and a privacy-preserving targeting mechanism in order to achieve relevance. We explicitly consider the possibility of payment of compensation to consumers directly for the piecemeal use of their private data. This variation raises the possibility of users "gaming the system", a problem for which we provide a novel, efficient cryptographic solution.

* Internal Accession Date Only
Approved for External Publication

DataBank: An Economics Based Privacy Preserving System for Distributing Relevant Advertising and Content

Rajan M. Lukose and Mark Lillibridge
HP Labs
1501 Page Mill Road
Palo Alto, CA 94304
{rajan.lukose, mark.lillibridge}@hp.com

ABSTRACT

A critical feature of successful new advertising models has been their reliance on knowledge of various types of personal user data in order to make advertisements relevant and useful. This has raised many concerns about privacy and control over personal data. The preservation of privacy would appear to be in direct conflict with the successful new advertising models, which depend on knowledge of personal user data.

Here we present a system and its associated advertising model that shows this need not be the case. DataBank is a system for the delivery of relevant advertising (and content more generally) while preserving the privacy of user data as much as possible. The system relies on an economic pricing mechanism, similar in spirit to economic approaches to spam, and a privacy-preserving targeting mechanism in order to achieve relevance. We explicitly consider the possibility of payment of compensation to consumers directly for the piecemeal use of their private data. This variation raises the possibility of users "gaming the system", a problem for which we provide a novel, efficient cryptographic solution.

Categories and Subject Descriptors

H.3.5 [**Information Systems Applications**]: Online Information Services—*Data sharing*

General Terms

Economics, Algorithms

Keywords

privacy, spam, pricing, advertising, filtering, recommendations, commitment

1. INTRODUCTION

Advertising has emerged as a successful and robust business model on the web that subsidizes the production of many valuable aggregations of content and a wide variety of services. One critical feature of the newer advertising models has been their reliance on knowledge of various types of personal user data in order to make advertisements relevant and useful. The simplest example is the successful sponsored-search business model in which the data is simply the search terms entered.

As online services accumulate increasing amounts of longitudinal user profile data in centralized stores in order, at least in part, to deliver more personalized advertisements, the issue of privacy has become more important. The preservation of privacy would appear to be in direct conflict with the new lucrative business models, which depend on knowledge of personal user data.

Here we present a system that shows that an effective advertising model need not violate privacy. DataBank is a system for the delivery of relevant advertising (and content more generally) while preserving the privacy of user data as much as possible. The system relies on an economic pricing mechanism, similar in spirit to economic approaches to email spam [9, 5], in order to achieve relevance. It is clear that increasing the cost of sending messages, especially advertisements, will make sent messages more relevant to the recipients. If this increase in relevance is not to occur by simply reducing the number of messages sent, means must be available to target messages only to recipients strongly interested in them. Accordingly, DataBank provides a mechanism that allows precisely targeting messages to users based on their behavior, while simultaneously preserving privacy as much as possible.

DataBank uses a very strong privacy model: all personal user data is kept locally on the user's computer. Only the personal data necessary to charge advertisers is sent to a DataBank server and then only when authorized by the user. The user can see exactly what information will be released in advance. Contrast our model with alternatives where users' personal data is accumulated on central servers, which are attractive targets for attackers.

We also explicitly consider the possibility of payment of compensation to consumers directly for the piecemeal use of their private data. This variation departs from most of the prior literature on economic approaches to information distribution. It raises a set of new questions regarding the set of economic equilibria that are likely to result. While a complete analysis of these economic considerations is underway and beyond the scope of this paper, we present heuristic arguments that demonstrate the existence of interesting equilibrium regimes. These regimes are made possible in part

by a novel, efficient cryptographic solution to the problem of users "gaming the system".

Our motivation in considering payments to users stems from the observation that user data (especially behavioral data) clearly has great economic value. Companies therefore have strong incentives to gather as much personal data as possible on users. This tendency need not necessarily be a concern to privacy advocates since there are some economic forces that motivate companies to be careful with the data they collect [2]. Nevertheless, technological solutions that allow some of the benefits of personalized, targeted, and relevant advertising, while preserving user control of their private data are worth exploring in our view.

Furthermore, we argue that some companies earn large economic rents from advertisers for the valuable services they provide based (in part) on their collection of user data. But user data also belongs to the users who may themselves wish to earn economic value from their own data. The DataBank architecture allows exploring these issues more deeply. Finally, given concerns about privacy, we believe that mechanisms such as DataBank may increase welfare for both advertisers and consumers by permitting and encouraging exchanges that would otherwise not occur.

The remainder of the paper proceeds as follows: In Section 2, we review related work and place the contributions of this paper into context. Section 3 presents an overview of the DataBank system and architecture, introducing local-client profile storage and querying, the charging system implemented in our prototype, and discusses the incentive issues that arise. Section 4 discusses the gaming problem that can arise when users are compensated for viewing ads, presents the options to ameliorate it, and explains a novel solution for a restricted class of ads. Section 5 presents a new and more efficient implementation of set commitment, a cryptographic primitive needed by our gaming problem solution. Finally, Section 6 concludes.

2. RELATED WORK

Prior research on economic approaches to solving the problem of spam email [9, 5] have demonstrated that the cost to the sender of the communications channel has a strong effect on the relevance of messages to recipients. Senders in these proposals are motivated to target messages to recipients, but the proposals typically leave unspecified the method by which messages can be targeted in practice. Our system implements the ability for senders to sort users into "types" (required by Zandt [9]); moreover, these types can be chosen so that they can be used to distinguish between recipients based on the value of a given message to them (required by Loder et al. [5]).

The DataBank architecture is designed to provide this capability without the wholesale release of real-time personal data to centralized data stores. This is made possible by bringing the queries to the data rather than the reverse, and takes advantage of low bandwidth costs to do so. In this regard, its operation is similar to that of the Shock system [6], which used the same technique to target messages while preserving privacy; Shock, however, relies on social pressure to prevent spamming, which is known not to work in large groups.

Commercial systems such as AllAdvantage.com have in the past attempted to compensate users for viewing advertising messages. These companies' business models failed in part because they did not have sufficient controls to restrict users gaming the system.[1] In this paper we analyze the gaming problem specific to compensating for clickthroughs, and provide a novel, practical cryptographic solution.

Our cryptographic solution is related to prior work that uses Merkle trees to commit to data sets [4, 8, 3]. However, because that work addresses applications such as time stamping and efficient distribution of certificate revocation information, it makes no attempt to preserve secrecy. Here we are concerned as well with the secrecy, including zero-knowledge, aspect of set commitment due to the privacy concerns of our system. Micali et al. [7] considers the same problem for database queries in general, but uses different techniques. Our method favors efficiency over query richness in order to meet our application needs.

3. THE DATABANK SYSTEM

DataBank is a messaging system which assumes the existence of advertisers, or content producers of any kind, who are willing to pay to reach consumers with their advertisements or content.

The DataBank system has user data privacy and control, message relevance, and ease of use as its three main objectives. Data privacy and control are achieved by an architecture that is, in our prototype implementation, device-centric. That is, user profile data, which is descriptive of users, is stored locally on client devices. Our prototype uses client PCs as the devices.

3.1 Architecture

Figure 1 diagrams the basic architecture of DataBank. There are three types of agents in the system: advertisers (or producers of content of any kind), a broker, and clients (associated with users). Each DataBank client device must be running the DataBank software. This software operates as a process running in the background that logs activity of various kinds to a database stored on the client and associated with the user. This client profile can include any information deemed useful. In our prototype software, the client logs URLs visited by the user, search engine keywords entered by the user, and software installed on the client. Each of these different kinds of data is stored as a separate type in the client database.

Messages from advertisers are sent to the broker. Each message contains a query and a hyperlink with some short descriptive text to an advertisement. Messages are thus small and can be on the order of 1 KB. The broker gathers up messages from the advertisers and periodically sends them, when requested, to clients. All clients receive all (new) messages asynchronously, which explains the broadcast layer in Figure 1. (Bandwidth requirements are discussed in more detail in Section 3.2).

Upon receipt of a message, the client runs the enclosed query against its local database. If the query is not satisfied by the database, then the message is discarded. If the query is satisfied, then the message is kept for future display to the user (the prototype client interface will be described later). Note that at this point, regardless of query satisfaction, the privacy of the user is maintained: neither the broker or any advertiser knows whether the query has been satisfied.

[1]See http://news.com.com/2100-1023-251949.html for more information.

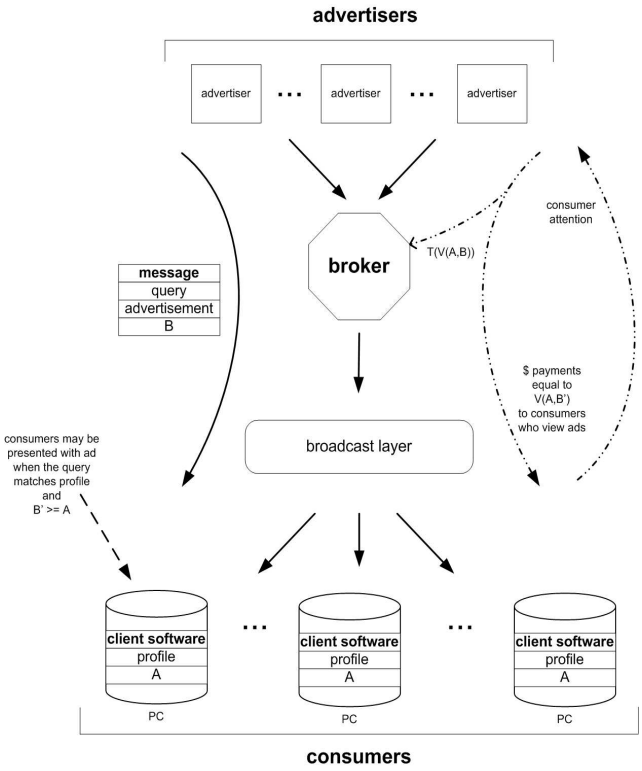

Figure 1: The DataBank system.

To make the discussion concrete, suppose Volvo wishes to target users looking to buy a safe SUV that are not already considering Volvo. Volvo accordingly might choose to send a message to users who have searched on the term "SUV safety" and have visited the domains "lexus.com" and "acura.com", but not the domain "volvo.com". The advertisement might consist of a web page describing the safety features of Volvo's new SUV. In this case, the query is easily expressed in DataBank's query language, and the message also contains some descriptive text describing the advertisement along with a URL to the full advertisement on the manufacturer's website.

Assuming a user's profile matches the query described, the user is notified through the client interface that a new message has arrived. If, and only if, the user clicks on the presented hyperlink to the advertiser's website, is it possible for the advertiser (and the broker) to know that the user has matched the query. Optionally, the user may request to view the query. The user may also simply discard the message without following the hyperlink as well.

3.2 Bandwidth

In the DataBank system, the privacy requirement places somewhat of a burden on the network, by broadcasting messages. As is often the case, the privacy requirement causes a tradeoff with efficiency. Nevertheless, the bandwidth requirements can be made modest and manageable.

The key issue is the average message size, and the rate at which new messages are created. In the DataBank system, we allow for targeting to occur based on reduced "factors" as well as raw data. These factors are essentially pre-defined semantic categories composed of lower level data elements (e.g., the "car shopper" factor can exist in the client and is a summarization of a more complex query based on URLs visited or searches done, etc.). Factors reduce message size.

We estimate that a meaningful message can be as small as 300 bytes. Even if there are 10,000 new messages/day, the bandwidth load on the client would only be 3 MB, which is quite manageable, especially using a polite download scheme over a broadband connection. On the server side, bandwidth costs at current market prices are such that the cost per month to support roughly 10 million users at 1,000 messages per day is approximately $20K.

3.3 Payments

The system as described so far is susceptible to spam. We define spam as an (untargetted) message that is likely to be irrelevant to a very large fraction of its recipients. Up to this point, there is no significant cost to a spammer for sending a message with a query that trivially match every user profile.

In the same spirit as some approaches to the problem of spam email [9, 5], DataBank employs an economic approach that makes the sending of messages by advertisers costly. While there are a variety of ways to do this, each with their own consequences on participant incentives, here we focus on the method shown in Figure 1.

The DataBank client allows the user to set an individual threshold "ask" price A. Advertisers, as a part of message, must additionally set a "bid" price B that represents the amount they are willing to pay per clickthrough for that advertisement. In general, the broker may transform the bid B to $B' = f(B)$ before forwarding the altered message to the client. In addition to query satisfaction, only if $B' \geq A$, will the hyperlink to the advertisement be presented to the user. Thus, if a user does not wish to see many advertisements, she can set her threshold A very high (making it less likely that $B' \geq A$). Alternatively, she can set her threshold very low if she is more inclined to see advertisements.

Given that per-click payments on search engines such as Google can be often be several US dollars (and sometimes tens of dollars; the average is approximately $0.60 [1]), DataBank allows that a portion of the funds an advertiser pays can go directly to the consumer. If a user chooses to click on the hyperlink to the advertisement, she is entitled to a payment of $V(A, B)$ once she clicks. To compensate the broker for various costs (administrative, maintenance, legal, reputational, etc.), the broker receives a payment $T(V(A, B))$.

As an example of the choice of payment functions, let us choose $B' = f(B) = B$, $V(A, B) = A$, and $T(V(A, B)) = B - A$. In this case, when a user with threshold A qualifies for a message (i.e., her user profile satisfies the message's query) whose bid price $B' \geq A$, and furthermore clicks on the presented link to the advertisement, she is paid exactly A. The advertiser is charged B, and the broker is paid $B-A$.

Note that in this simple case, there is a chance that $B = A$ so the broker receives 0. However, this problem can easily be solved, for example, by using a bid transformation function in which the broker is paid a fixed fraction t of B, and the value of B transmitted with the message to the client is actually the remainder $B' = (1-t)B$. Then, in cases where $B' \geq A$, the broker receives the fixed fraction upon clickthrough, in addition to the difference between the reduced effective bid and ask prices.

3.4 Incentives

To summarize, under the scheme in Figure 1, advertisers are able to target messages to users with great precision based on their locally stored, private profile data. Users are able to receive messages, while maintaining control of their private data.

The threshold pricing scheme described in the previous section is designed to create the right types of incentives for advertisers and consumers, so that advertisers reach those who are most likely to want to receive their messages, and consumers receive messages that are most likely to be relevant to them, while maintaining control over their private data, and furthermore, monetizing their own data.

Advertisers naturally seek to minimize their costs and so would like to lower B. Consumers, depending on the opportunity cost of their participation in the system, may either seek to lower their asking price A (in order to make money using the system, regardless of the intrinsic information value of messages), or raise it to an appropriate level in order receive useful information and compensation.

A formal economic analysis of the resulting equilibrium (including laboratory experiments with human subjects) is underway, and we provide here a heuristic analysis. For users with relatively high opportunity costs for using the system, we assume the cost of gaming the system will not be worth their effort and time. However, for those with low opportunity costs, gaming the system may be an attractive proposition. Those users will be likely to lower their threshold in the hopes of gathering many messages on which to click.

However, if this behavior is prevalent, advertisers in equilibrium will not send messages subject to such gaming since they will not be receiving enough for their investment. The question then becomes whether, in equilibrium, any messages are sent at all. This problem resembles the problem of click fraud in current (highly successful) advertising business models such as Google's AdWords and AdSense programs. These business models survive because (apparently) the problem of click fraud can be kept under control; advertisers are aware of click fraud, and discount what they are willing to pay to account for it.

We argue that messages will be sent in equilibrium as long as targeting is effective in providing relevant, informationally rich messages to enough participants. The automatically created profiles, which can in principle include any data available to the client, are designed to allow for high-quality targeting. Like the attention bond mechanism of Van Alstyne et al. [5], the ability of senders to craft usefully targeted messages is key.

It is also, of course, critical to make gaming difficult or even impossible. It turns out that it can be made strictly impossible, but only for a class of messages which are characterized by the fact that their associated queries are unpredictable by users. For messages outside that class, various techniques can be used to ameliorate the problem, but we believe it cannot be stopped completely. How to deal with gaming is discussed more fully in Section 4

3.5 Interface

Our prototype implementation of DataBank is fully functioning and runs on Microsoft Windows clients. The server component of the system (the broker) runs on an open source stack. Figure 2 shows the two components of the client interface. The lower portion of the figure shows the always visible indicator in the systray portion of the desktop which summarizes the amount available to the user for clickthroughs. The upper portion of the figure shows the main client interface which is a small application window that appears when the user clicks on the systray indicator.

Figure 2: The DataBank client interfaces.

As shown in the figure, the user can easily set her threshold for the minimal clickthrough payment she is willing to consider. In this case, the threshold is set to $1.01 and the advertiser is willing to pay $1.25 for a clickthrough. (In this version of the prototype, the broker receives no payment, and $V(A, B) = B$.) The ad message is targeted in accordance with the example discussed in Section 3.1. Thus, only users who matched that specific query would have received this message, and possible payment. Users can easily raise their threshold using the slider if they wish to receive fewer and more relevant messages. Through configuration options, users are also easily able to delete their local profile database as well as pause and restart logging.

Payments are implemented using the PayPal Web Services API, using email addresses that are part of the configuration settings of the client software. The server (broker) interface is not shown. It straightforwardly allows the creation of messages including bid price setting and query construction.

4. COMPENSATING USERS

Because advertisers are charged an appreciable amount for each ad shown and ads may be precisely targeted, we expect DataBank ads to be highly relevant to the users they are shown to. Thus, users may wish to voluntarily watch

DataBank ads for free, although they may demand a higher level of relevance (i.e., set a higher threshold) than advertisers would prefer.

If we can offer users incentives for watching ads, we can increase the benefit of DataBank to both users and advertisers. To see this, consider a user that is willing to watch ads costing their advertiser $10 for free. He has learned from experience that cheaper ads are not relevant enough for him. Now consider an advertiser that wishes to show an advertisement worth $8 (i.e., the advertiser expects to make $8 on average from showing the ad) to the user. While this advertisement is likely not relevant enough to the user that he is willing to watch it for free, he may well be willing to watch it for a $1 incentive. By charging the advertiser $7, $1 of which is given to the user, we can make both the advertiser and the user $1 better off than if no user incentive was available, leaving the advertisement unseen. Incentives need not be in cash; for example, $1 worth of frequent-flier miles on the airline of the user's choice could be offered instead.

However, offering incentives for watching ads if not done with great care can lead to severe gaming problems that destroy the utility of the DataBank system. If users can earn valuable incentives through particular behaviors that are targeted by incentive paying ads, they will be motivated to game the system by performing those behaviors (perhaps in an automated manner) solely to earn the incentives; this breaks the linkage between those behaviors and the users' receptiveness to the ads' messages. While the previous advertisement may be worth $8 when shown to one of the users it is targeted at, if there are nine gamers for every targeted user, then the ad will be worth only $0.80 per clickthrough, which is too little to pay the incentive from.

We know of three basic ways to address this gaming problem: choosing incentive types that do not appeal to gamers, capping incentives, and unanticipatable incentives.

4.1 Unappealing incentives

One approach is to pay incentives only to charities of the user's choice rather than to the user themselves. While this reduces the value of each incentive somewhat, it disproportionately reduces the motivation to game the system: people who are okay with stealing from advertisers are unlikely to value donations highly. Likewise, poor people or people with little disposable income but a lot of spare time (e.g., teenagers) may be highly motivated to game a system for cash, but would have little interest in benefiting charities. Requiring a credit card with a reasonable limit or other form of income verification in order to receive incentives may also help reduce gaming.

Restricting incentives to discounts on the products being advertised is another approach: a gamer, by definition, is not interested in the products being advertised. Discounts cannot be used for advertisements that are not expected to result in product purchases such as branding messages. Like payments to charity, discounts are less effective incentives than straightforward cash payments for watching ads. In the case of discounts, this stems from users' uncertainty of their true value. Users' uncertainty has two sources. First, users are uncertain whether the original pre-discounted price advertised is fair. For example, an Acme widget with an advertised "retail" price of $100 plus a $20 discount might well be readily available elsewhere for $70, meaning that the "discount" is really nothing of the sort. Requiring advertisers to provide pricing guarantees ("if you can find an Acme widget elsewhere in the next five days cheaper than $100, we will send you the price difference plus $5") may ameliorate this source of uncertainty.

Second, the value of a $20 discount depends on how likely the user is to actually buy the product in question. If there is a 10% chance he will buy, it is worth $2, but if there is only a 1% chance he will buy, it is only worth $0.20. This is especially problematic because the user needs to decide before he sees the advertisement in question, and hence before he knows what product the discount is for. The user's threshold cannot be used to bound the buying probability: the amount the advertiser is forced to pay for showing an ad ensures that the advertiser expects to make at least that amount; however, because the expected profit is the profit of a sale times the probability of the user buying the product, the buying probability is relatively unconstrained by the user's threshold. The user is thus forced to assume an average buying probability when weighing the value of offered discounts.

This information asymmetry—advertisers have a better idea of the buying probability—makes discounts inefficient. Advertisers of rarely desired, but very profitable products pay less for the same of amount of incentive effect, leading them to advertise disproportionately, lowering the average probability and thus the incentive power of a given discount below what it should be. Note that discounts unlike other incentives do not contribute to making ads more relevant because they are paid only to highly interested users—discouraging spam requires charging advertisers per *uninterested* user.

It may be possible to address the second source of uncertainty by providing an estimated buying probability with each ad based on the conversion rate (conversions per clickthrough) to date or of a sample population. Unless a sample population is used, early estimates will be too low because of the time lag between viewing an ad and purchasing the advertised product. This suggests an advertiser may wish to use a high bid price to reach enough users in a randomly chosen subset of users to establish a high buying probability then switch to a lower bid price plus a (now fairly valued) discount to reach the remaining users. If these estimates can be made accurate enough, users may be willing to accept a lower bid price for ads with high estimated buying probability because of these ads' demonstrated relevance to their targeted users.

4.2 Capping incentives

Another approach to dealing with the gaming problem is to limit the total amount of incentives available to each person in a given period. If the total amount of incentives available is equal to the expected amount of incentives earned by a non-cheating user plus an amount small enough that it is not worth gaming the system to get, only rarely targeted users that earn substantially less than the cap will be motivated to game the system. Lowering the cap further will decrease the number of these users at the cost of decreasing the effectiveness of incentives for more frequently targeted users. A simpler formulation where every ad has the same incentive is to pay every user who watches at least some minimal number of ads per time period a fixed amount.

It is important that the cap be per person, not per user, so that gamers cannot evade the cap by each creating many

virtual user accounts. We believe requiring each user to have a unique non-P.O. box residential address in order to collect incentive checks will mostly prevent this. Unfortunately, while adopting a cap will reduce gaming, we do not believe it by itself will reduce gaming enough to make paying incentives viable.

4.3 Unexpected incentives

The final approach to dealing with the gaming problem is to limit incentives to ads targeting unanticipatable groups of users; that is, almost no one would have anticipated that ad's target behaviors would be the target of an ad with an incentive. We have in mind here the often flashy ads that inaugurate an advertising campaign for a new product rather than the bread-and-butter ads that repeatedly target new parents for baby strollers. Each product could be advertised this way only once. Care must be taken to ensure that the target behavior includes sufficient negative features (behaviors that targeted users must not have performed) that gamers attempting to earn incentives by performing (or at least claiming to have performed) all the behaviors they can think of will not be targeted.

The key to making this approach work is preventing users from copying rewarded behaviors. Because users are not all online and receptive to ads at the same time, it is very hard to avoid there being be a sizable time period between when the first users are rewarded for viewing a given ad and when the incentive expires. In the absence of a mechanism to prevent it, early users will be strongly tempted to tell their friends how to qualify for the same incentive they received. The design of such a mechanism is made harder by the fact that we cannot trust DataBank clients because users can modify them, by the fact that it must preserve users' privacy, and by the fact that our privacy model requires that rewarded users can determine—possibly with some hacking—what behaviors they were rewarded for.

We have devised a mechanism that solves this problem using a cryptographic primitive we call *set commitment*. Informally, set commitment allows one party called a commiter to commit to the contents of a set to a second party called a verifier without revealing anything about the set and then later reveal the (non-)membership of selected elements to the second party, again without revealing anything else about the set. Which elements to reveal information about are chosen by the commiter and it is computationally infeasible for him to lie to the verifier. Set commitment is a generalization of regular cryptographic commitment, which involves a first party committing to a value to a second party and then later optionally revealing it. Micali et al. [7] first describe general set commitment under the name zero-knowledge sets, giving an inefficient implementation. We use instead a more limited, but substantially more efficient implementation we independently discovered; see section 5 for a description of our set commitment scheme.

The idea behind our mechanism is to divide advertising into two phases. First, each client commits to its current user profile to a DataBank server by a deadline, treating each present queryable-about feature of the user profile as an element of the to-be-committed-to set. Second, ads accompanied by their targeting queries and offered incentives are sent to all the clients. When a user wants to accept the incentive associated with an ad that targets them, their client reveals to the DataBank server only the features (positive or negative) of their committed-to user profile required to prove that it satisfied that ad's query. If the DataBank server is convinced and that user's client's commitment was made by the deadline, then the incentive is paid.

The user's privacy is protected by the secrecy properties of set commitment: absolutely no information is released unless the user claims an incentive, and even then only the minimum necessary to claim the incentive is released. If desired, the client can explain exactly what information must be released in order to claim an incentive. The commitment properties, in turn, protect the system from gaming: in order to claim an incentive, a user must have qualified for it before the deadline and hence before anyone other than the advertiser saw the associated query; copying the behavior of other users who received the incentive afterwards will not qualify a user for the incentive. No amount of client hacking can change this.

In practice, of course, this process would be repeated with overlapping phases; each ad would query a version of a user's profile that had been previously committed to. The need to represent user profiles as a set of features and to make queries satisfiable by a small number of positive and negative features (the cost of revealing is proportional to the number of elements revealed) somewhat limits the expressibility of queries under this mechanism even with careful design.

5. OUR COMMITMENT SCHEME

We count each demonstration that a element is (is not) in a committed-to set as a separate membership (non-membership) proof. Thus, revealing that 1 and 2 are in a set but 3 is not requires two membership and one non-membership proofs.

Micali et al. [7] describe how to implement a set commitment scheme that allows generating as many membership and non-membership proofs as desired; however, its commitment step essentially requires three modular exponentiations and one hash per element in the set to be committed to. This is too slow for our purposes because we wish to use sets containing tens of thousands or even hundreds of thousands of elements. For example, we would like to include each URL visited by the user in the last month as a separate feature.

Accordingly, we use a more limited, but more efficient set commitment method of our own devising. In exchange for limiting in advance the number of non-membership proofs that can be generated, we can reduce the commitment cost to essentially one hash per element in the set to be committed to plus a number of hashes dependent on the number of non-membership proofs desired.

We illustrate our scheme using a series of increasingly complicated schemes, leading up to the full scheme. Like Micali et al., we assume the set to be committed to, call it S, consists of L-bit values for some value of L. This can be arranged by cryptographically hashing each of the original elements and then keeping only the first L bits of each hash; sufficiently large values of L ensure that the assignment of two features of interest to the same L-bit value is vanishingly small.

5.1 Commitment without secrecy

For our first scheme, in the commitment step we encode the features of set S in the leaves of a complete height L prefix Merkle tree. See figure 3 for an example where $L = 3$

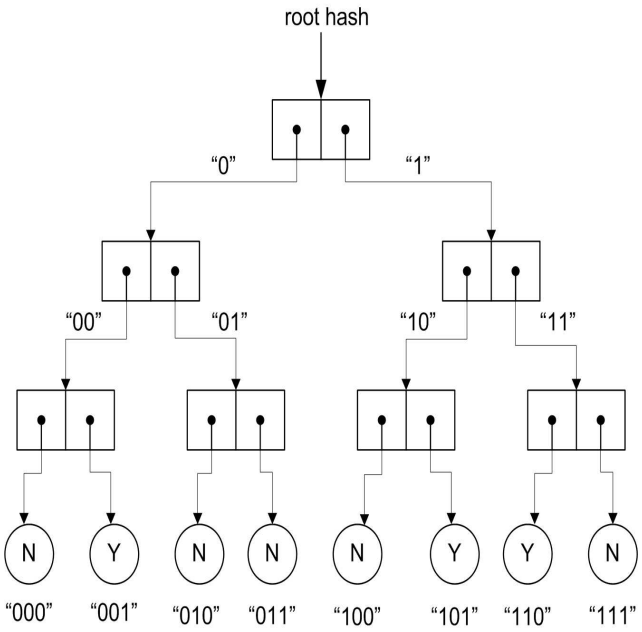

Figure 3: A height 3 complete prefix Merkle tree encoding $\{001_2, 101_2, 110_2\}$; all pointers contain the hash of the node they point to.

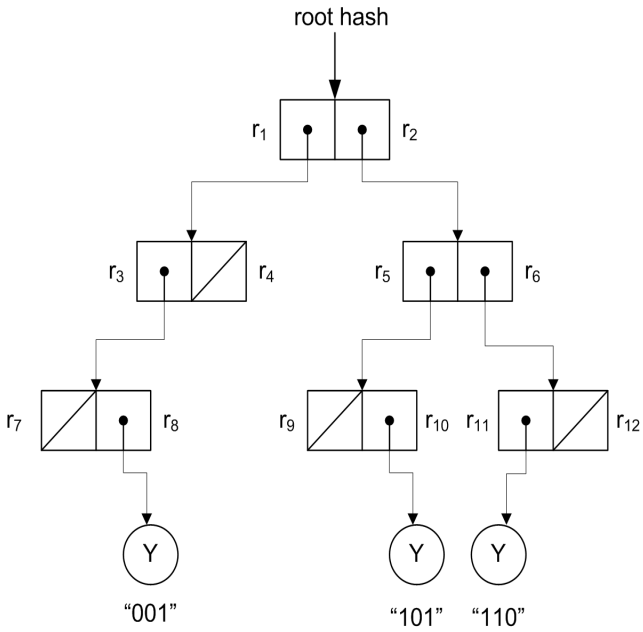

Figure 4: An alternate encoding replacing "N"-only subtrees with null; pointer i is blinded using the random value r_i.

and $S = \{001_2, 101_2, 110_2\}$. A prefix tree of height L is a binary tree of height $L+1$ which maps L-bit strings to leaf nodes in the following manner: Start at the root node. Each time you come to a node (including the root node), go left if the next bit of the string (starting from the left) is a 0 and right if the next bit of the string is a 1. In figure 3, "000" is mapped to the first (leftmost) leaf node node, "001" to the second leaf node, "010" to the third leaf node, and so on. We encode which elements are present in the set by indicating in each leaf node whether or not the L-bit string that maps to it is in the set.

Because the generated tree is a Merkle tree, each of the pointer fields holds the cryptographic hash of the contents of the node its pointer points to rather than the location of the node it points to. We finish the commitment step by providing the cryptographic hash of the root node to the second party. Any collision-free hash function may be used for our scheme; we currently use MD5, but may switch to SHA-2. For the simplified analysis of this paper, we will assume the cryptographic hash function used is equivalent to a random oracle. Revealment is easy: to reveal element e's membership status simply provide the leaf node e is mapped to and all its ancestors to the second party. The second party verifies such statements by checking that the revealed path is a valid path (i.e., each node's hash agrees with the pointer to it) from the previously revealed root hash.

The properties of Merkle trees ensure that the first party cannot change his mind between the commitment and revealment steps about the tree. Doing so requires producing two different nodes with the same hash, which is computationally infeasible. Our encoding of element membership ensures that the same tree cannot be used to show the element is both in and not in the set at different times. Thus, our first scheme provides the desired commitment properties. It, however, does not preserve secrecy and is terribly inefficient for large values of L as it requires over 2^L hashes, which is likely exponentially larger than the size of set S.

The lack of secrecy comes from the fact that the second party may be able to determine the contents of unrevealed nodes whose hashes are known by guessing their contents then checking if their hashes match. For example, if the first party reveals the fact that "001" is present, the second party will be able to determine that "000" is not present using the pointer to its leaf node contained in the revealed immediate parent of the "001" leaf node.

5.2 Secrecy for membership proofs only

Our second scheme corrects both of these deficiencies at the cost of not allowing non-membership proofs. We correct the efficiency problem by replacing subtrees containing only "N" leaves by the special value null. To obtain secrecy, we switch to *blinded pointers*. A blinded pointer is a pointer that can only point to one node and that can only be followed with its creator's aid. Figure 4 shows the result of applying the second scheme to our example set S. We denote pointers to null by slashes.

A blinded pointer's value is computed by taking the hash of the byte sequence composed of the contents of the node it points to (or the empty string if it points to null) preceded by a fixed-length large random number. Each blind pointer is associated with a large random number; these random numbers are not considered to be part of any node. If each random number has as many bits as a cryptographic hash then (under the random oracle model) the blind pointer value by itself reveals nothing about what node it may point to even if infinite competition resources are available. The node, if any, a blind pointer points to can be revealed by reveal-

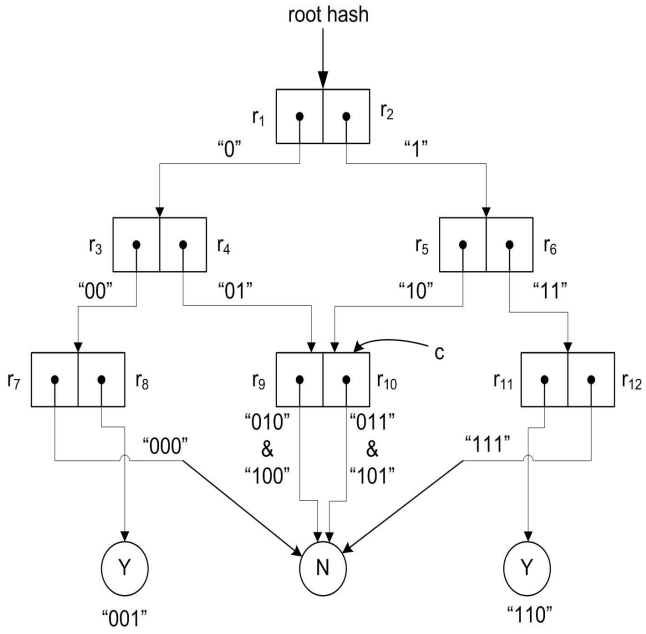

Figure 5: A blinded optimized prefix Merkle DAG that reuses nodes and encodes $\{001_2, 110_2\}$.

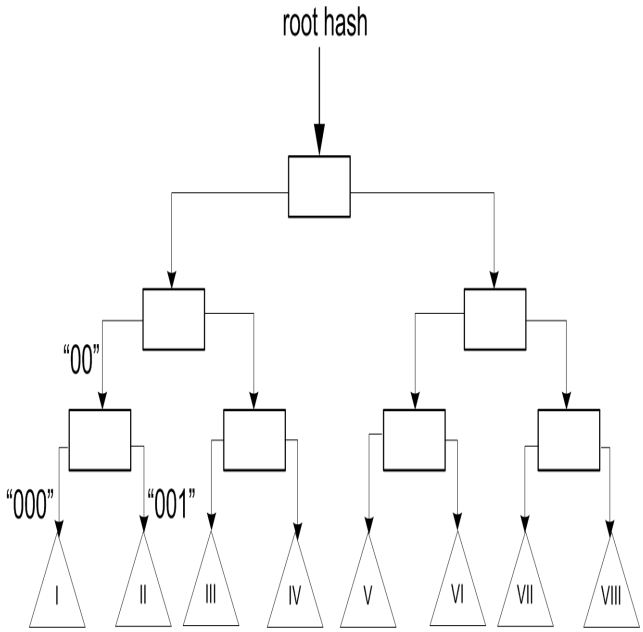

Figure 6: A blinded optimized Merkle DAG that reuses nodes within regions; node and region details are omitted. Here $K = 3$, resulting in 8 regions, I-VIII.

ing its associated random number; because of the properties of cryptographic hash functions, a given blind pointer can be revealed to point to only one thing. In essence, a blind pointer is just an ordinary commitment to the hash of the node it points to.

Revealment now involves also revealing the associated random numbers of the pointers that lead to the leaf nodes associated with the elements whose membership status is being revealed; no other random numbers are revealed. For example, the membership proof for 101_2 using figure 4 is

$$(p_1, p_2),\ r_2,\ (p_5, p_6),\ r_5,\ (p_9, p_{10}),\ r_{10},\ \text{“Y”}$$

where p_i is the contents of pointer field i. This hides all information about the nodes pointed to, but not included in the proof; e.g., in the above $p_6 = \text{hash}(r_6; p_{11}; p_{12})$ appears random because r_6 is unknown.

Because the path taken to the leaf node associated with an element in the set is independent of what other elements are in the set, this hiding means that revealing membership reveals nothing beyond what is intentionally revealed. This property, unfortunately, does not hold for revealing non-membership because the location of null pointers depends on the membership status of many elements.

5.3 Allowing one non-membership proof

Our next scheme improves on the previous scheme by allowing at most one non-membership proof without compromising secrecy; as before, any number of membership proofs can also be generated without leaking information. The key idea is to reuse nodes with no “Y” descendants as much as possible. See figure 5 for the result of applying this optimization to the example set $\{001_2, 110_2\}$ (S without the element 101_2 to simplify the figure). Notice that figure 5 has only one “N” leaf node and only one node, labeled “c”, with exactly 2 descendants, both of which are “N” leaf nodes.

In general, this scheme uses at most L more nodes than the previous scheme, one for each possible complete “N”-only-descendants subtree. This scheme produces a directed acyclic graph (DAG) rather than a tree; if the resulting DAG is unfolded by duplicating reused nodes to form a tree and the pointers are unblinded, the result is our first scheme.

The commitment properties hold as before (Merkle DAGs work like Merkle trees in this regard). Because we are using blinded pointers, secrecy holds as long as the second party cannot tell which revealed non-leaf nodes are reused in the DAG: under this condition, the revealed paths are indistinguishable from those that would have been produced from a version of the DAG that reuses only the “N” leaf node. The only way for the second party to discover that a non-leaf node has been reused is for it to see two proofs that together demonstrate that the node is reachable in two different ways from the root node.

Because no membership proof’s path goes through a reused node, the only way secrecy can be lost is for two different non-membership proofs to be generated that share a reused non-leaf node. For example, in figure 5, the non-membership proofs for “011” and “100” share the interior node labeled “c”. The second party could deduce from this information that “010” and “011” are also not present because reused nodes under our scheme never have “Y” descendants. Thus, if we restrict ourselves to never generating more than one non-membership proof per committed-to set, we will never leak information.

5.4 The full scheme

For our full scheme, we generalize the idea of the previous scheme to allow a small number of non-membership proofs to be issued. We divide up the prefix tree we generate into

regions and reuse nodes only within a region. By dividing up the generated tree into regions based on the first K bits of each string, we get 2^K regions as illustrated in figure 6. Each region has $L - K + 1$ reused nodes that encode each of the possible subtrees containing no "Y" leaf node descendants. Each region is thus the result of applying the at most one non-membership proof scheme to the elements in the set that start with that region's prefix after stripping that region's prefix from those elements.

Under this construction, two non-membership proofs can only have a reused node in common if they are for two elements whose first K bits are the same. Thus, at the cost of at most $2^K(L - K + 1)$ extra hashes, the first party can issue up to 2^K non-membership proofs so long as each is for an element with a different K-bit prefix.

The restriction on which non-membership proofs can be issued can be made less onerous by randomly assigning elements to each region. To do this, a random permutation is applied to all the elements before they are added to the set. The permutation used is published with the root hash.

For example, to commit to a set $\{000000_2, 000001_2\}$, the first party would choose a random permutation P and commit to the set $\{P(000000_2), P(000001_2)\}$ using the multiple region scheme; the first party publishes the resulting root hash and P. To prove that 100000_2 and 100001_2 are not in the original set, he shows proofs that $P(100000_2)$ and $P(100001_2)$ are not in the committed-to set. He can do this safely as long as $P(100000_2)$ and $P(100001_2)$ differ in their first K bits, which will happen with probability $1 - 2^{-K}$. Note that the probability he can do this is independent of the other contents of his set, so whether or not he can provide a proof does not leak extra information.

More generally, the probability that the first party can safely produce N non-membership proofs is approximately $e^{\frac{-N^2}{2^{K+1}}}$ (cf. the birthday paradox). If we want no more than an ϵ probability that the first party cannot produce N non-membership proofs, then we need at least $\frac{N^2}{2\epsilon}$ regions since $\ln 1-\epsilon \geq -\epsilon$ for $0 < \epsilon < 1$ by Taylor expansion.

Thus, if we want to be able to issue N non-membership proofs with probability $1-\epsilon$, our scheme requires $|S| + \frac{LN^2}{2\epsilon}$ hashes compared with Micali et al.'s $3|S|$ modular exponentiations and $|S|$ hashes.[2] Note that ϵ is not a security parameter, which needs to be extremely small. For DataBank, it determines an upper bound on what fraction of targeted users are missed; a value of $1/100$ thus might be appropriate. Because we do not expect to pay many unexpected incentives per commitment period and we do not expect individual queries to need to refer to many negative features, we should be able to use quite small values of N. As an example, $L = 32$, $N = 6$, and $\epsilon = 1/100$ yields 57,600 extra hashes. Verifying a proof under our scheme requires only L hashes whereas Micali et al. require $3(L-1)$ modular exponentiations and L hashes.

If the queries and hence the non-membership proofs that may be required are known in advance, the revealment step can be made fail-proof: the DataBank server can choose a permutation and value of K that definitely maps each of the elements whose non-membership may need to be established to a different region. The DataBank server then requests clients to commit using the calculated permutation and value of K. This does provide a little advance information about the queries, but almost certainly not enough to matter.

[2]These calculations ignore operations on non-reused interior nodes on the assumption that $|S| \gg L$.

6. CONCLUSION

The success of advertising business models on the web has highlighted the tension between privacy and advertising relevance. In this paper we presented DataBank, a system designed to simultaneously achieve as much privacy and control for user data as possible, while allowing for some of the benefits of personalization based on private data. Within our system, the benefits depend on advertisers sending messages that are likely to be relevant and useful to recipients.

We presented an economic approach, similar in spirit to prior proposed spam solutions, which ensure that senders will send relevant messages in equilibrium by making the channel for the messages costly to senders. The DataBank architecture allows this to be done with little privacy and data control loss on the part of users, and provides a practical way for senders to have access, through DataBank queries, to the raw data necessary to make messages relevant.

We also examined the possibility for users in the system to capture for themselves some of the economic value of their own private data. Many methods for doing this are possible, but we paid particular attention to one of the most difficult methods, in which some of the per click amount charged to advertisers is shared with users. We analyzed the resulting substantial gaming issues, and provided a novel, efficient and practical cryptographic solution that solves the gaming problem for a class of messages, and we described various ways to make the gaming problem manageable in the other cases.

7. ACKNOWLEDGMENTS

We thank Joshua Tyler, Stephen Sorkin, and Norm Jouppi for helping to develop the ideas in this work. We especially thank Tyler for his server implementation, and Sorkin for his implementation of our set commitment scheme.

8. REFERENCES

[1] *http://blog.searchenginewatch.com/blog/050106-095817*, 2005.

[2] J. A. Deighton. *Digital Anonymity and the Law — Tensions and Dimensions*, chapter 6, pages 137–146. T.M.C. Asser Press, 2003.

[3] P. T. Devanbu, M. Gertz, C. Martel, and S. G. Stubblebine. Authentic third-party data publication. In *IFIP Workshop on Database Security*, pages 101–112, 2000.

[4] S. Haber and W. S. Stornetta. How to time-stamp a digital document. *Journal of Cryptography*, 3(2):99–111, 1991.

[5] T. Loder, M. V. Alstyne, and R. Walsh. An economic answer to unsolicited communication. In *ACM Conference on Electronic Commerce (EC'04)*, 2004.

[6] R. M. Lukose, E. Adar, J. R. Tyler, and C. Sengupta. Shock: communicating with computational messages and automatic private profiles. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 291–300, New York, NY, USA, 2003. ACM Press.

[7] S. Micali, M. Rabin, and J. Kilian. Zero-knowledge sets. In *Proceedings of the 44th IEEE Symposium on Foundations of Computer Science (FOCS '03)*, pages 80–91. IEEE Computer Society Press, 2003.

[8] M. Naor and K. Nissim. Certificate revocation and certificate update. In *Proceedings 7th USENIX Security Symposium (San Antonio, Texas)*, Jan 1998.

[9] T. V. Zandt. Information overload in a network of targeted communication. *Rand Journal of Economics*, 35:542–560, 2004.