Privacy-Preserving Global Customization

(Extended Abstract)

Robert M. Arlein   Ben Jai   Markus Jakobsson   Fabian Monrose   Michael K. Reiter

Information Sciences Research Center
Bell Labs, Lucent Technologies
Murray Hill, NJ, USA

ABSTRACT

We present an architecture for global customization of web content, by which a web site can customize content for each visitor based on the activities undertaken by the same user on other, unrelated sites. Our architecture distinguishes itself in the privacy mechanisms it provides: each user controls what information a merchant can learn about her activities at other merchants, and each merchant controls to what other merchants the information it contributes is revealed. To achieve this we introduce novel data protection mechanisms for merchants and users. We further describe aspects of a prototype implementation of our architecture.

1. INTRODUCTION

Mass customization refers to the creation of a customized experience for online buyers by using technology that responds to their individual requirements and interests [11].[1] Customization typically employs data mining and/or collaborative filtering to predict content that is likely to be of interest to that visitor, and presentation of customized content to the visitor at opportune moments. Customization can be particularly effective when the user identifies herself explicitly to the web site. In this case, customization can be much more "accurate", in the sense that the site can employ the specific user's past browsing and purchasing history at that site to predict what content will be most effective for this user.

In this paper we describe the design and implementation of a novel infrastructure for *global* customization. Our in-frastructure enables global profiles of each user's behavior to be maintained, so that a merchant can customize content for a user based on that user's activities even at *other* merchants. At the same time, however, our infrastructure is privacy preserving, in the sense that users and merchants can control how information about them is shared. Specifically, our infrastructure enables each user to control which of her information can be gathered together in a profile, and does so with natural extensions to the user's browsing experience. It also enables each merchant to specify which other merchants can learn the information that it contributes to a profile or other information derived from it. This is a considerable divergence from approaches today, in which data protection models are lacking.

The mechanisms by which our system protects merchant and user privacy are novel themselves. For users, we employ the abstraction of a *persona* in which a user conducts web activity. A user can have many personae, with the property that only the user's activities undertaken while in a given persona can be linked in a profile. This gives the user a convenient and natural way to partition information about herself into profiles that she can selectively reveal. For example, a user may create one persona for work, one for recreation, and one for seeking medical information. For merchants, we introduce a powerful protection model based on *tainting*, which offers each merchant fine-grained control over which merchants can access the records it contributes about its customers or information derived from those records. This gives merchants the ability to specify different gradations of access control for partners, competitors, and others.

To make our discussion concrete, consider the following example of the type of capabilities we envision. Suppose a user purchases a ticket to Egypt at a travel web site. Later the consumer visits an online bookstore, which learns of the consumer's interests in travel and Egypt by means of the infrastructure proposed here. The site thus customizes its pages based on this information, highlighting books about the pyramids, tours and travel in Egypt, etc. When the consumer visits an online electronics store, the entry page highlights their new Arabic-to-English electronic pocket translator, and so on. However, at any point the user can switch to a different profile that reflects nothing about these activities, and so this information will not be conveyed to sites the user subsequently visits. Moreover, the book store can specify that records it contributes to the profile (e.g., that the user bought books about Egyptian art) not be made avail-

---

[1] "Customization" is sometimes called "personalization", though personalization also conveys the meaning of web content that the user can explicitly configure. For example, a user might create a personalized web page at a site by telling the site which stock quotes to display whenever the user visits. Here we are primarily concerned with content that a site predicts the user will like based on information inferred about the user, rather than by explicit user instruction.

able to other book stores, since these competing book stores could use this information to gain this user as a customer.

This extended abstract reports on the major components of an architecture that enables such interactions. Section 2 discusses related work. The goals of our system and the interfaces it offers users and merchants are described in Section 3. Section 4 describes the infrastructure we have built for achieving these goals. Section 5 describes how personae are managed in the system, and Section 6 details merchant privacy protections. We discuss applications of our approach to business-to-business transactions in Section 7.

2. RELATED WORK

Global customization, by which a user's web history is shared across many merchant sites, is practiced today in several forms. A predominant form of such global customization is "ad networks" such as DoubleClick. In this form, information about a visitor's activities at a merchant site is passed to DoubleClick via image hypertext links in the merchant's page. In response to these requests, DoubleClick returns banner advertisements customized to these activities. This customization is "global" in that this information is collected into a profile for the user (or more precisely, the browser) that is used to customize ads for the same user on her future visits to DoubleClick-enabled sites. Recently, more ambitious sharing of global user web profiles has been developed by companies like Engage (www.engage.com), Angara (www.angara.com) and I-behavior (www.i-behavior.com; also see [12]). These companies profile users—Engage and Angara using an opt-out approach and I-behavior using an opt-in one—and provide targeted information to merchants about a user for the purposes of customization. Our architecture improves upon these approaches by providing support for users and merchants to specify policies that limit who can obtain information they contribute.

To date, the predominant attempt to provide users more control over how their information is shared on the web is the Platform for Privacy Preferences (P3P). P3P is a protocol to be executed between a web site and a *user agent* (e.g., running in a web browser or proxy server) in order to automatically negotiate the disclosure of user information as a function of the information being requested and the site's privacy policy. Reagle and Cranor [13] suggest that some P3P implementations will support a data repository from which user information is served, and that this repository might be populated by explicit user configuration or by insertion of data by a web site. They further suggest that a user agent might implement multiple personae per user that each provides different information to sites, an abstraction that we adopt in our system. Our system differs from P3P mainly by shifting focus from solely the user to both the user and the merchant. First, our system anticipates the storage of user records in a repository outside the user's control (as deployed systems suggest is the trend today) and provides an implementation of the persona abstraction for this setting. Second, it implements merchant privacy controls, which P3P does not address. Third, by separating the storage of persona profiles from any information linking those profiles, our system does not offer a single location that, if compromised, would reveal both all profiles for the user and their relation to that user.

A system that exploits P3P explicitly for global customization is due to Cingil et al. [4]. This work describes an architecture for creating a global web profile for a user by employing an HTTP proxy that monitors the HTTP traffic to and from that user's browser. The user then employs P3P to negotiate the disclosure of information from this profile to merchant sites. In our approach, merchants contribute information to user profiles. This should yield more accurate and meaningful profiles than what might be automatically gleaned by an HTTP proxy. It also underlines the need to address the privacy concerns that merchants have regarding the information they contribute to profiles, which our system attempts to address. Another system employing an HTTP proxy to gather global profiles for customization (but without using P3P or the persona abstraction) is that of Predictive Networks (www.predictivenetworks.com). Our comparison to that system is similar.

Also related to our work are electronic wallets, such as Microsoft Passport (www.passport.com) and the Java wallet (java.sun.com/products/commerce), that offer possibilities for global customization similar to those we propose here. Wallets vary with respect to what information they retain about user activities, whether that information can be contributed by merchants or only the user, and to what extent wallets share this information with participating merchants. However, to the extent that wallets do retain behavioral information—at least they often retain receipts for purchases, for example—such wallets pose a privacy risk to both users and merchants. From the user perspective, these wallets hold identifying information for the user in conjunction with any behavioral information, and so stored behavioral profiles are not anonymous. Moreover, to the extent that behavioral information is conveyed to merchants, merchants do not have sufficient opportunity to specify data protection policies about how information they contribute is to be shared with others. These privacy risks have been cited as a major tension between wallet vendors and both online merchants and users; e.g., see [3]. Our infrastructure proposed here, while offering a wallet-like interface to the user, is an attempt to provide a better alternative.

Finally, web personae by which a user can undertake web activities are analogous to psuedonymous email addresses, or "nyms", for email (e.g., [10, 7]). Users post to newsgroups or send emails under a nym in a way that recipients cannot correlate multiple nyms as being the same user. Our personae offer a similar abstraction in the context of web activity and profiling thereof.

3. GOALS AND ABSTRACTIONS

In this section we describe the goals of our system, and the main abstractions it offers to merchants and users. We note immediately that our goals do *not* include limiting the collection of information that already takes place today on the web. Preventing data collection by technical means is the topic of numerous other research and commercial projects in anonymous or pseudonymous web access (e.g., [14, 15, 6, 7]). Our work complements this research by providing a means of controlled information sharing that is compatible with existing web infrastructure and even with anonymous web access, e.g., as implemented by the aforementioned anonymizing systems.[2] Similarly, our goals do not include preventing

---

[2]Most anonymizing systems can be configured to remove HTTP cookies from traffic between the browser and web sites. Since our prototype uses cookies, it is compatible with

various privacy attacks that, e.g., enable a web site to directly observe a user's activity at other web sites (e.g., [5]). The same measures and precautions against such attacks should be applied by users of our system.

A second important omission is that our goals do *not* include preventing merchants from sharing information outside our system. Rather than trying to force the adoption of our system by eliminating alternatives to it, our intention is to offer a more publicly acceptable and valuable infrastructure to enable this sharing. As a result, the threats we consider do not admit collaborative misbehavior by merchants to convey more information among themselves than is allowed by our policies. Merchants could always convey that information outside our system, and indeed risk being detected if they misuse our system for that purpose. (Auditing compliance with our policies is discussed below.) That said, we do try to ensure that if merchants share data outside our system, then the mechanisms our system introduces are of no help in their doing so.

Within the constraints outlined above, our first goal is to enable each user to partition behavioral record-keeping by merchants into several profiles that are unlinkable to those components that possess them, and to control which profile is exposed to each merchant. This goal requires us to separate storage of profiles from the ability to link those profiles to a single user, and to our knowledge this distinguishes our work from prior art. For merchants that contribute information to profiles, our protection goal is to offer a sufficiently rich protection model for the merchant to control what other merchants can benefit from those records (also a distinguishing feature). Finally, our goals include achieving the prior without changing existing web infrastructure; in particular, we do not require the use of custom client-side software (in contrast to, e.g., the Java wallet).

While we strive to *prevent* abuse of our system whenever possible, in some cases we resort to auditing to *detect* (and thus discourage) forms of abuse that cannot be inherently prevented, or charging models to *motivate* merchants to behave appropriately. For example, since merchants sharing data outside our infrastructure cannot be prevented, perhaps merchants should be periodically audited by an organization like TRUSTe (www.truste.org) or BBBOnline (www.bbbonline.org) as a condition of using our system. Other behavior that can be audited is the accuracy of records that merchants contribute to a profile, though doing so requires a different form of audit, i.e., active probing. To conduct this form of audit, an auditing agency could play the role of a user who visits the merchant and conducts some transaction. Afterward, the records the merchant contributed can be examined for accuracy. To motivate merchants to contribute records that are useful to other merchants, the pricing of our system can be structured in a way that a merchant is issued a credit when other merchants use a record contributed by this merchant.

3.1 The user's perspective

The main abstraction that our system presents to the user is that of having (possibly) multiple personae. A persona is

a role in which the user engages in web activity; natural personae may be "work", "entertainment", "medical", "shopping", "investing", etc. The relevant feature of a persona is that activities undertaken by the user while acting in a given persona can be linked and profiled across sites. So, if a user visits two different sites under his "work" persona, then information about his activities undertaken at each site will be available to the other site, provided that both sites allow this. However, if he visits a site under his "work" persona, then he need not fear that his "entertainment" activities will become known to that site.

Because it is intrinsically difficult to prevent the correlation of two personae of the same user at a single site—e.g., the two personae could be linked based on IP address, an HTTP cookie, or possibly even browsing behavior—by default our system allows a merchant to read the profile of only one persona per user. This is achieved by granting read credentials to a merchant for only that persona; for a different persona employed by the user on a subsequent visit to that site, the merchant will not be given credentials to read that persona's profile. In the parlance of [2], merchants are thus subject to a *Chinese wall* policy in the sense that the profiles for a user's personae constitute a conflict-of-interest class. However, the user may override this policy, e.g., allowing a merchant to contribute records to one of her personae even if it cannot read that persona. A user may even choose to allow a merchant to read from multiple of her personae, particularly if she is convinced that the merchant will not be able to link them—e.g., if she uses different browsers from different computers when acting under these different personae. (We note, however, that our architecture supports the use of the same personae from multiple computers.)

Users can configure personae on various parameters, which will be described in Section 5. A user selects a persona when a site requests a persona and one has not already been selected by the user for this browsing session. Our scheme is opt-in, i.e., this interface is not presented to the user unless she previously enrolled to receive persona requests, and at any point she can disable a persona and later re-enable it via a simple interface. It is important that users be able to understand the policies associated with personae, and to easily switch between personae when appropriate. Our design is constructed with this as a primary concern.

3.2 The merchant's perspective

Commerce servers are often built using database-driven templates that enable dynamic web page creation (e.g., [18]). In this approach, web page templates are written in a template language and stored in the web server file system. This template language offers primitives for posing queries to databases, performing computation, and rendering HTML. Thus, when the web page is requested, the web page template is interpreted to render a web page based on information retrieved from databases; see Figure 1.

Our system adds to this picture another "database" per merchant, called the Global Customization Engine (GCE). Conceptually this serves as another database that web page templates can query. However, rather than being a database of only local information, it interacts with remote components of our infrastructure to obtain web history information about (the persona of) a visitor to this site and to contribute information about this visitor. Web page templates can query the GCE for information about the visitor, and they

---

these anonymizing systems only when they are configured to not remove cookies. If cookies are not available for use, either due to an anonymizing system or because the user has disabled their use in her browser, then our prototype has no effect and is invisible to her.
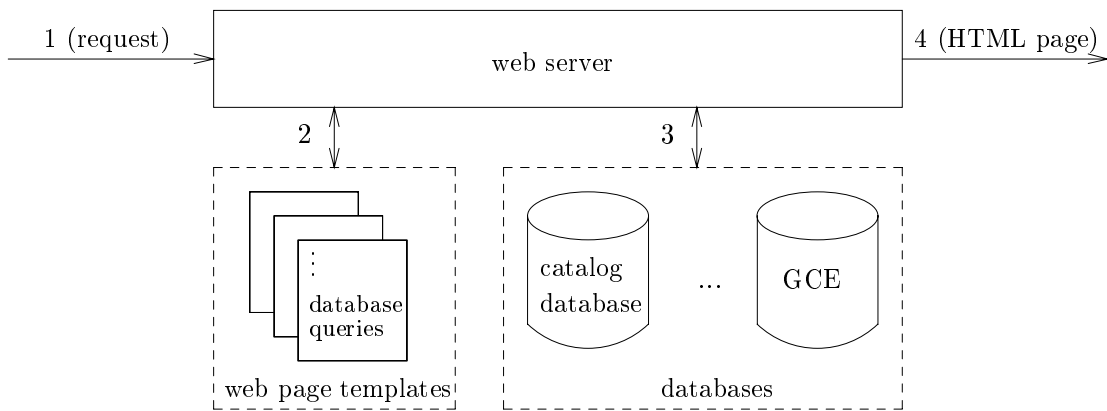
Figure 1: Simplified architecture of a typical commerce server, with GCE added

or other components (e.g., CGI scripts) can insert records about this persona at the GCE. The GCE may propagate these records to other components of our architecture, which may eventually propagate these records to other merchants this user visits.

The interface between the merchant site and the GCE enables the merchant site to register an identifier for the visitor along with a *persona access credential* (PAC) that is passed to the merchant site if the user's persona management policy allows. From then onward (until the PAC expires), web page templates can query the GCE using the chosen identifier. The GCE uses the PAC to retrieve information from our infrastructure about the persona associated with the corresponding PAC. The PAC also enables the merchant to contribute information about the visitor to our infrastructure. When the merchant site inserts records at the GCE, it specifies access control information that constrains what other merchants can read these records or records derived from them. This data protection model will be covered in Section 6.

We have implemented a GCE, integrated it with a commerce server (iMerchant Pro 2.0; see www.ihtml.com), and set up merchant servers in this configuration for our own testing purposes. This commerce server supports a web page template language called iHTML, via which web pages pose queries to the GCE. In our implementation, the merchant registers a PAC with a customer identifier that it also sets as an HTTP cookie in the user's browser for the current browsing session. When the site gets an HTTP request from that user, it can pass the associated cookie to the GCE to obtain information about the (persona of the) visitor. We have designed our GCE so that it can be run on a separate host from the merchant site and therefore requires minimal changes to existing sites.

4. INFRASTRUCTURE

In this section we give an overview of the infrastructure we have developed to support the interfaces described in Section 3. The primary and thus far unmentioned components are the following.

- **Persona server**: A *persona server* resides in the network to support the management of user personae and the issuance of PACs. Each user who employs our sys-

tem holds an account at the persona server. This account allows the user to create new personae and manage policies for existing personae. Users must trust the persona server to accurately enforce the policies the user specifies for her personae, and to not disclose relationships between personae and users to merchants. In order to scale, the persona server might be implemented as a single *virtual* server with one domain name. This name would be dynamically mapped to an actual persona server depending on a range of criteria, including the proximity of the server to the client, the current load and availability of servers, etc. Techniques for implementing virtual servers and the dynamic mapping of DNS queries to actual servers is known and practiced today; Akamai (www.akamai.com) is a particularly visible example.

- **Profile database**: A *profile database*, or PDB, contains records inserted by merchants about different personae. There can be numerous, unrelated PDBs in our system. A merchant chooses the PDBs to which it inserts records as those it trusts to enforce the data protection policies that it specifies; PDB support for merchant data protection will be described in Section 6. Users must trust the PDBs of the merchants to which it provides PACs to limit merchants to the forms of access specified in those PACs (Section 5). However, since users may not be aware of the PDBs a merchant uses, this trust will need to be gained with, e.g., the assistance of an auditing body (see Section 3).

We emphasize that a user's persona server should be separate from the merchants the user visits and the profile databases they use: since the persona server stores the correspondences between personae and users, joining the persona server with profile databases would enable construction of a profile per user—as opposed to per persona. We thus envision the persona server being established as a privacy preserving site devoted to this purpose. We anticipate PDBs being offered by service providers, particularly as a value-added feature for commerce server hosting.

The type of data that merchants insert into PDBs should be limited to information about *what* a persona (i.e., user) did while at their web sites. In particular, it should exclude information that could be used to link two personae,

such as the IP address from which the user visited or any other identifying information like email address. Note that our decision to disallow multiple personae to be read by any merchant by default takes away incentive to do otherwise: a single merchant, even if in theory it could link two personae to the same user, will not be given PACs to read data for both personae. This restriction on the type of data merchants insert thus primarily serves to prevent PDBs from linking personae associated with the same user.

Implementation issues. We have implemented a protocol by which a merchant site requests a PAC for a persona from the persona server, the persona server issues that PAC, and the merchant uses it (via its GCE) to read or insert information about the persona to a PDB. We omit description of this protocol in this extended abstract, but a few comments are in order. First, this protocol requires user input only in the case that there is no current persona for the user. Second, this protocol employs HTTP tools (e.g., redirection, cookies) in a way similar to the protocol by which Microsoft's Passport service (www.passport.com) issues credentials. This is not surprising given that both systems are designed for unmodified client browsers, albeit for somewhat different tasks. The limitations of HTTP and surrounding infrastructure result in limitations of our protocol similar to some recently pointed out for Passport [8]. Where possible, we have taken steps as suggested in [8] in our implementation to address these limitations.

5. PERSONAE MANAGEMENT

As already discussed, personae are the basic tool by which users partition their behaviors into profiles. The main challenge to implementing personae is to enable the user to easily configure her personae with the desired policies for protecting her privacy, and in some cases to make policy decisions for the user so that managing personae is not a burden.

5.1 Persona configuration

The policies that describe how personae are managed and how PACs are distributed can significantly impact how a user's data is shared. Some of these policies, and how they can be configured in our prototype, are described below.

Rights conveyed with PACs. A PAC granted to a merchant enables that merchant to access the information in a PDB associated with the persona named in this PAC. With one exception described below, by default a PAC conveys *read* rights, which enable the merchant to read records in the PDB associated with the persona, and *insert* rights, which enable the merchant to insert new records about that persona. However, a user could grant only one of these to a merchant. For example, a user might grant a site only read access if she does not want her activities at that site added to her profile. She might grant only insert access if she does not want the site she is visiting to learn her other profiled data, but she is comfortable with that site adding data to her profile. A third type of access can be granted: *delete* rights, which enable the merchant to delete records associated with the persona from the PDB. Delete rights make it possible to set up a monitoring site that users can visit to review the information stored about their personae in a PDB and delete records of their choosing.

Exposure of multiple personae at one merchant. As already discussed in Section 3.1, granting PACs to a merchant with read rights for two different personae of the same user potentially enables the merchant to "merge" the personae profiles, even if the PACs are sent to that merchant in two different sessions. For this reason, we have adopted the default policy that a merchant site be granted read rights to only one persona per user, namely the first persona under which the user visits the site. This policy, however, is limiting in certain cases. For example, many web sites may naturally be visited by the same user in different personae, such as search engines and portal sites that serve as general "launch points" for content regardless of what type of content is sought. Allowing only one persona to be read by each of these sites may limit the amount of customization that site can perform.

Duration of a persona as a default. When a user selects a persona in which to browse, that persona can become the default, or "current", persona for some period of time, in order to minimize interruptions in the user's browsing experience. A configurable parameter of a persona is the length of this duration. The default setting for this parameter is the duration of the browsing session, i.e., until the user closes her browser. Other alternatives are a specified time period (e.g., 30 minutes), or simply to not make the persona a default at all. The default persona can be switched by the user explicitly. A persona, even if the default, will not be made readable to a site if that site previously was sent a PAC containing read rights for a different persona of the same user.

Duration of PACs. The duration for which a PAC (and the access rights it conveys) is valid can have significant ramifications to user privacy. On one end of the spectrum, a PAC granting read access that is valid indefinitely enables the site that receives it to monitor this persona arbitrarily far into the future. On the other end of the spectrum, a PAC may be limited for use only within a very tight time frame, perhaps only for a minute or so before it has to be renewed. Here the tradeoff involves the additional overhead of frequent renewals, but the benefit to the user is fine-grained control over the duration for which she can be monitored (in the case of read access) or data about her can be added (in the case of insert access). In our prototype implementation we have adopted a short duration period for PACs by default, in order to better protect the user's privacy.

5.2 PAC format

Persona access credentials (PACs) are granted by a persona server to a merchant to enable the merchant to read, insert and/or delete records for this persona. A PAC is a structure containing the following fields:

1. *An identifier for the merchant to which the PAC was issued.* This identifier is used by PDBs to verify that the merchant presenting a PAC is the same merchant to which that PAC was granted. In our prototype, this identifier is the public key that PDBs use to authenticate requests from the merchant. This public key must be conveyed to the persona server within a certificate that is appended to the PAC request and signed by a certification authority known to the persona server.

2. *An expiration time.* This time is calculated as a function of the PAC duration as described in Section 5.1.

3. *Access rights.* By default, these include both read and insert permissions, or at most insert permission if this PAC is being issued to a merchant to which a PAC containing read permission for another persona of the same user was previously issued. However, the user may choose a different configuration of access rights, possibly including the additional delete permission.

4. *A digital signature on the above items.* When a persona is created at the persona server, the server creates a new public key pair for it. (See motivation below.) That private key is used to sign all PACs for that persona.

5. *The persona public key.* The public key matching the private key used to sign the PAC is sent with the PAC. This public key serves as the long-term identifier for the persona.

A PDB verifies a PAC accompanying a merchant request by first verifying its signature using the public key contained in the PAC (i.e., the persona public key), and verifying that the PAC has not expired. It then compares the access rights granted in the PAC to the request that the merchant is making, to determine whether it should grant this request. If the request is allowed, the PDB performs the request on the data associated with the persona public key; i.e., this public key is used as the index for a persona's data.

There are several items worth noting about this construction. First, our design includes a separate public key pair per persona in anticipation of distributed persona server implementations. This will enable fine-grained audit and control over which persona server replicas can grant PACs for each persona, by controlling and monitoring to which replicas each persona private key is communicated. Second, the persona public key need not be certified in any way. If the merchant forges a PAC using a different public key, then it is merely posing queries to a nonexistent persona for that user. Third, the use of a digital signature and verifying public key in the PAC eliminates any need for a shared key between the PDB and persona server. The persona server need not know to which PDB(s), if any, the PAC will eventually be presented.

In our current implementation, persona public keys are RSA keys [16] with 1024-bit moduli. All cryptographic operations in our persona server and PDB implementations are performed using the Cryptolib library [9], version 1.2. PAC signature and verification take 8 ms and less than 1 ms, respectively, on the Sun Ultra 250 (296 MHz UltraSPARC-II CPU) that is our persona server. A commercial implementation might further employ a hardware crypto-accelerator.

6. DATA SHARING AMONG MERCHANTS

Data sharing among merchants takes place by merchants inserting records into, and reading records from, a PDB via their respective GCEs. For the purposes of this section, we denote the merchant who inserted the record a by $\mathsf{merchant}(a)$, and the persona (i.e., the persona public key; see Section 5.2) to which the record pertains as $\mathsf{persona}(a)$. We do not distinguish between the merchant site and its GCE in this section.

Just as users have privacy concerns that must be addressed in our system, so do merchants. Specifically, a merchant may not want to insert records into the PDB if a competing merchant can use this information, directly or indirectly, to tailor content to the same user if she happens to visit the competing merchant. Thus, for our system to be adopted by merchants, it must protect the information that they insert into the system.

6.1 A tainting data protection model

The data protection model that we have adopted for this task is based on information flow models, specifically tainting. Intuitively, one datum in the system *taints* another if the value of the second was influenced by the value of the first. A tainting model enforces the policy that if a taints a', then a' can be used only in ways that a has been authorized to be used. So, for example, if the owner of a specified that it not be disclosed, then a' cannot be disclosed either. The general idea for using tainting to protect merchant data in our system is that for each record a that a merchant inserts into the PDB, the merchant specifies sets of other merchants to which it will allow that record, or anything that record taints, to flow. So, for example, if a merchant reads a and uses it to customize pages for a user, and then the merchant inserts a record a' based on the user's subsequent behavior (e.g., perhaps the user bought what the merchant displayed), then a' can be read only by merchants that the merchant who wrote a allows it to.

In order to make this viable in practice, however, the model must be considerably refined. The primary reason is that if a data item taints records arbitrarily far in the future by default, this will prevent much data sharing among merchants, usually unnecessarily. For example, consider the scenario outlined in Section 1, in which a user purchases travel to Egypt and consequently is offered, and buys, books about pyramids from an online bookstore. Now suppose the user visits an online home furnishings store, which offers the user a reading lamp because it learns of the user's interest in reading from the records inserted by the book store. In this example, it would typically be unnecessary that records inserted by the home furnishings store, indicating the purchase of a reading lamp, be withheld from other travel stores that the user visits merely because records inserted by the first travel store are contained in their causal history.

We thus enrich the model by requiring a merchant to specify *taint classes* for each record a that it inserts. Abstractly, the merchant specifies a sequence of sets $\mathsf{class}_a[0]$, $\mathsf{class}_a[1]$, ..., $\mathsf{class}_a[\mathsf{str}(a)+1]$, where each $\mathsf{class}_a[i]$ is a subset of merchants, $\mathsf{class}_a[i] \subseteq \mathsf{class}_a[i+1]$, $\mathsf{class}_a[\mathsf{str}(a)+1]$ is the universe of all merchants, and $\mathsf{str}(a)$ is a nonnegative integer called the *taint strength* of a. Intuitively, if merchant m is not in $\mathsf{class}_a[i]$, then it is not allowed to read records derived from a by a sequence of i or fewer derivations. More precisely, suppose we define a relation $\rightarrow$ as follows:

DEFINITION 6.1. $a \rightarrow a'$ *if and only if*

$$\mathsf{merchant}(a) \neq \mathsf{merchant}(a') \quad \wedge \qquad (1)$$
$$\mathsf{persona}(a) = \mathsf{persona}(a') \quad \wedge \qquad (2)$$
$$\mathsf{merchant}(a) \; read \; a' \; before \; inserting \; a \qquad (3)$$

Now consider the directed acyclic graph formed by the $\rightarrow$ relation, i.e., where nodes are records and edges correspond to the $\rightarrow$ relation. For records a, a' and merchant m, if

$m \notin \mathsf{class}_a[i]$ and there is a path of length i or less from a' to a, then m cannot read a'.

We anticipate that a merchant will make use of this model by specifying sets of merchants when it registers with a PDB and then referring to those sets to construct the taint classes for records it inserts. For example, a merchant might designate a set M_{partners} of partner merchants with whom it is willing to share data generously, and a set $M_{\mathsf{noncompetitors}}$ of merchants that are neither partners nor competitors. Then, when it inserts a record a, the merchant might specify $\mathsf{str}(a) = 1$, $\mathsf{class}_a[0] = M_{\mathsf{partners}}$ and $\mathsf{class}_a[1] = M_{\mathsf{partners}} \cup M_{\mathsf{noncompetitors}}$. That is, only partners can read record a, and only partners and noncompetitors can read records $a' \rightarrow a$ (and only if $\mathsf{merchant}(a')$ consents). In particular, competitors of $\mathsf{merchant}(a)$ can read neither.

Our algorithm to enforce the policy expressed by taint classes is as follows. Stored with each record a is (i) an integer value called the *accumulated taint strength* of a, denoted $\mathsf{ats}(a)$; (ii) the sets $\mathsf{class}_a[1]$, ..., $\mathsf{class}_a[\mathsf{str}(a)]$; and (iii) (pointers to) the records a' such that $a \rightarrow a'$. When a record a is inserted, $\mathsf{ats}(a)$ is computed as

$$\mathsf{ats}(a) = \max\{\mathsf{str}(a), \max_{a':a \rightarrow a'}\{\mathsf{ats}(a')\} - 1\}$$

To determine whether a merchant m can read a, the PDB executes a breadth-first search from a in the graph defined by $\rightarrow$, truncating each descending traversal once when it encounters a record a' where $\mathsf{ats}(a')$ is less than the current depth in the search. For each record a' visited at depth d in this traversal, m is allowed to read a only if $m \in \mathsf{class}_{a'}[d]$ (or $d > \mathsf{str}(a')$).

To insert a record a with out-degree B (i.e., there are B records a' such that $a \rightarrow a'$), the computation required is $O(B)$. Determining whether a merchant can read a record takes $O(E(\log R + \log M))$ time if there are a total of M merchants, R records for this persona, and E edges among these records. In practice, however, the computation time will be much less for reasonable taint strengths. In particular, if a maximum taint strength per record were imposed, then the breadth-first traversal will stop by the depth of that strength. As described above, we anticipate that the sets comprising the taint classes for a record will be previously specified sets that categorize merchants relative to the inserting merchant. In this case, a record requires storage of only $O(B + C)$ pointers over and above the (one-time) storage of these merchant categories if there are C categories.

In our approach a merchant can change taint classes for a record even after inserting that record. However, to support changes that increase the taint strength of the record, it would be necessary for each record a to store pointers to all records a' such that $a' \rightarrow a$. Then, if the merchant changed the taint classes of a record a in a way that increases $\mathsf{str}(a)$, the PDB would recompute $\mathsf{ats}(a)$ and perform a depth-first traversal to depth $\mathsf{ats}(a)$ on the DAG defined on the inverse of $\rightarrow$, starting at a. For each node a' visited in this traversal, $\mathsf{ats}(a')$ would be updated if necessary.

To further improve efficiency and minimize unnecessary tainting, we are currently evaluating variations on our tainting model that "expire" taint. One such alternative is, when a merchant inserts a record a, to save pointers $a \rightarrow a'$ only to a fixed number of records a' most recently read by $\mathsf{merchant}(a)$. In this way, a record a' will eventually no longer taint records written by a merchant, if the merchant does not read a' again. In a second approach, each merchant could specify a set of "downgrading" merchants so that a merchant m can read a' if for each path from a' to a of length i where $m \notin \mathsf{class}_a[i]$, there is a record inserted by a downgrading merchant, as specified by $\mathsf{merchant}(a)$, on that path. Intuitively, $\mathsf{merchant}(a)$ might specify a merchant to be downgrading if the business of that merchant is so different from $\mathsf{merchant}(a)$'s that records it writes are deemed to be of no assistance to competitors of $\mathsf{merchant}(a)$.

6.2 Reading records

The records that a merchant reads is a primary factor in determining the taint properties of records that merchant inserts. In order to minimize unnecessary tainting, it is important that merchants read only records that are directly relevant to the customization decisions they make. We have thus designed a read interface for the PDB that makes it possible for merchants to target these records.

The PDB interface for reading records supports two types of operations. The first operation, here called create_list, takes as its arguments a PAC and a *scoring function* specified by the merchant. The scoring function f accepts as input a single record and returns a floating point value, called a *score*. Intuitively, for a record a, the score $f(a)$ indicates a's relevance to the customization decision that the merchant must make, as determined by the scoring function f. For example, a reasonable scoring function might return higher scores for more recent records, records that indicate large purchases by the visitor, or records that match the merchant's inventory well. In our current implementation, the scoring function is a Java class file that that the merchant administrators must craft, and that is required to implement a function with no side effects (i.e., no network communication, disk accesses, etc.). For convenience, we provide a set of prefabricated scoring functions that merchants can extend or use directly.

The create_list operation applies the scoring function f to all records to which the merchant has access for the persona indicated by the PAC. The return value from create_list is a reference L to a linked list of records sorted by descending scores, stored at the PDB. Importantly, invoking the operation create_list does *not* "count" as reading records, since the reference L that it returns does not indicate information about the content of records, their scores, or even how many records are in the resulting linked list stored at the PDB.

The only operation available to the merchant using the reference L is to invoke next(L). This operation initially returns the record at the head of the list, and when successively invoked it returns the next record in the linked list. Each record returned to the merchant is marked as having been read by the merchant, for the purpose of determining the records a such that $a' \rightarrow a$ for the records a' the merchant inserts. The merchant can sample the first few records of L to determine whether they suit the merchant's needs. If so, these can be used to customize content for the visitor. If not, the merchant site may form a new list by invoking create_list with a different scoring function. We expect that this interface will require the merchant to read very few records per visitor in order to customize its content, thereby limiting unnecessary tainting.

6.3 On accessing multiple PDBs

As described in Section 4, we allow multiple PDBs in our system, and further allow a single merchant to subscribe

to multiple PDBs as it chooses. It is thus possible that a record at one PDB will be tainted by a record at another PDB. More precisely, when inserting a record a to a PDB D, the merchant's GCE propagates to D a reference to each record a' at another PDB such that $a \rightarrow a'$. The graph traversals in the algorithms of Section 6.1 may then require communication across PDBs to complete. If a needed PDB is unreachable, our algorithms respond conservatively: e.g., in the case of determining whether a merchant can read a record, if the PDB at which a necessary record a' resides is unavailable, then the merchant is disallowed.

We note that placing responsibility on merchant GCEs to propagate this taint information poses minimal risk to the enforcement of tainting policies. First, there is little motivation for a merchant to modify its GCE to suppress the fact that $a' \rightarrow a$ when writing a record a'; doing so merely decreases the degree to which a' is tainted. Second, the fact that $\mathsf{merchant}(a')$ read a means that $\mathsf{merchant}(a') \in \mathsf{class}_a[0]$. That is, $\mathsf{merchant}(a)$ already trusts $\mathsf{merchant}(a')$ with a, and so trusting $\mathsf{merchant}(a')$ to propagate the fact that $a' \rightarrow a$ extends this trust minimally. Third, since the PDB storing a maintains the time at which a was read, and the PDB storing a' similarly records the time at which a' was inserted, such suppression is readily detected in an audit involving both PDBs. We thus believe that communicating records outside our system would be a less risky approach to violating our tainting model, consistent with our goals (see Section 3). If suppression of cross-PDB taint information is nevertheless deemed at risk in our architecture, this can be offset by the deployment of GCEs engineered to resist tampering by the merchant (e.g., [1]) or, in the limit, eliminated by restricting each merchant to use a single PDB.

7. APPLICATIONS TO B2B

The design of our system was influenced by our focus on the business-to-consumer market. For example, this is manifested in our goal of working with unmodified client browsers, where it is known that relying on user installation of new software can be a barrier to adoption. It is also manifested in our attention to user privacy. However, we believe that many principles underlying our system can be applied in certain business-to-business (B2B) settings.

One application of our design in B2B settings is in so-called "ScenarioNets", which is a model of interaction to which some B2B e-markets are evolving. Seybold et al. [17, pp. 36–39] define a ScenarioNet to be "a customer- and project-specific set of interrelated tasks that can be performed across web sites and suppliers to accomplish a specific outcome". The importance of "customer- and project-specific" is that the sequence of interrelated tasks may be so customized to the customer and project that it is not anticipated or directly supported by a vertical or horizontal e-market. Seybold et al. suggest supporting ScenarioNets by providing a way for the customer to carry the context of previously completed tasks from one web site to the next, so that already-entered information and results of already-completed tasks are available to the next sites and applications in the sequence. We believe that the infrastructure described in this paper could support ScenarioNets in this way, where the user could employ a persona per sequence of tasks. Our infrastructure provides both the means for context to be carried from one step to the next and mechanisms to protect the sensitive information of both the user

and web sites involved in the sequence of tasks. And, in contrast to the support offered by GroupWare systems, our system need not be configured with advance knowledge of the sequence of related tasks.

Applying our techniques in B2B settings may also change certain design criteria we adopted when building our prototype. This is particularly so for the requirement to support unmodified client browsers, since certain B2B settings will be more amenable to the introduction of custom client software. In this case, it is possible for this client software to embody the persona server for this user, or even the PDB contents themselves (similar to suggestions in [13, 4]). However, the latter organization would centralize all data in a way that reveals a single profile for the user if this centralized store were compromised.

8. CONCLUSION

In this extended abstract we have introduced a system for supporting global customization. Our system enables profiles of user information to be maintained and accessed by merchants. Via the persona abstraction, users control what information is grouped into a profile, and can selectively enable a merchant to read one or more of these profiles. Our architecture introduces a *persona server* that assists users in managing their personae. Our architecture separates this from the *profile databases* at which profile information is stored, to eliminate any single point at which different profiles can be tied to the same user. Since merchants also have privacy concerns, our architecture offers a data protection model based on tainting, by which merchants can limit how the information they contribute can be exposed.

Acknowledgements

9. REFERENCES

[1] D. Aucsmith. Tamper-resistant software: An implementation. In *Proceedings of the First International Information Hiding Workshop*, Lecture Notes in Computer Science 1174, pages 317–333, 1997.

[2] D. F. C. Brewer and M. J. Nash. The Chinese wall security policy. In *Proceedings of the 1989 IEEE Symposium on Security and Privacy*, pages 206–214, 1989.

[3] K. Cassar, L. Graves, and R. Sterling. *Digital Wallets, Pursuing Dual Wallet Strategy Before Leverage is Lost*. Jupiter Strategic Planning Services/DCS99-14, February 1999.

[4] I. Cingil, A. Dogac, and A. Azgin. A broader approach to personalization. *Communications of the ACM* 43(8):136–141, August 2000.

[5] E. W. Felten, D. Balfanz, D. Dean, and D. S. Wallach. Web spoofing: An Internet con game. In *Proceedings of the 20th National Information Systems Security Conference*, October 1997.

[6] E. Gabber, P. B. Gibbons, D. M. Kristol, Y. Matias and A. Mayer. On secure and pseudonymous client-relationships with multiple servers. *ACM*

Transactions on Information and System Security 2(4):390–415, November 1999.

[7] I. Goldberg and A. Shostak. Freedom Network 1.0 architecture. Manuscript, November 1999. Available from www.freedom.net/info/freedompapers/index.html.

[8] D. P. Kormann and A. D. Rubin. Risks of the Passport single signon protocol. In *Proceedings of the 9th International World Wide Web Conference*, May 2000.

[9] J. B. Lacy, D. P. Mitchell, and W. M. Schell. CryptoLib: Cryptography in software. In *Proceedings of the 4th USENIX Security Workshop*, pages 1–17, October 1993.

[10] D. Mazières and F. Kaashoek. The design, implementation, and operation of an email pseudonym server. In *Proceedings of the 5th ACM Conference on Computer and Communication Security*, pages 27–36, November 1998.

[11] J. Nelson. *Mass-Customization Marketing: Maximizing Value of Customers*. IDC Bulletin #17726, December 1998.

[12] P. Parker. Direct marketing guru founds new venture targeting e-commerce. *InternetNews.com*, March 16, 2000.

[13] J. Reagle and L. F. Cranor. The platform for privacy preferences. *Communications of the ACM* 42(2):48–55, February 1999.

[14] M. Reed, P. Syverson, and D. Goldschlag. Anonymous connections and onion routing. *IEEE Journal on Selected Areas in Communication* 16(4):482–494, May 1998.

[15] M. K. Reiter and A. D. Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security* 1(1):66–92, November 1998.

[16] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, Feb. 1978.

[17] P. B. Seybold, M. M. Frey, S. E. Aldrich and J. E. W. Miller. *Understanding the B2B and E-Market Landscape*. Customers.com Focused Research Collection, Patricia Seybold Group, Inc., 2000.

[18] G. W. Treese and L. C. Stewart. *Designing Systems for Internet Commerce*. Addison-Wesley, Reading, Massachusetts, 1998.