

Experimental Demonstration of a Hybrid Privacy-Preserving Recommender System

Esma Aïmeur and Gilles Brassard
Université de Montréal
Département d’informatique et de R.O
C.P. 6128, Succursale Centre-Ville
Montréal (QC), H3C 3J7 Canada
{aimeur, brassard}@iro.umontreal.ca

José M. Fernandez
École Polytechnique de Montréal
Département de génie informatique
C.P. 6079, Succursale Centre-Ville
Montréal (QC), H3C 3A7 Canada
jose.fernandez@polymtl.ca

Flavien Serge Mani Onana and Zbigniew Rakowski
Université de Montréal
Département d’informatique et de R.O
C.P. 6128, Succursale Centre-Ville
Montréal (QC), H3C 3J7 Canada
{manionaf, rakowskz}@iro.umontreal.ca

Abstract

Recommender systems enable merchants to assist customers in finding products that best satisfy their needs. Unfortunately, current recommender systems suffer from various privacy-protection vulnerabilities. We report on the first experimental realization of a theoretical framework called ALAMBIC, which we had previously put forth to protect the privacy of customers and the commercial interests of merchants. Our system is a hybrid recommender that combines content-based, demographic and collaborative filtering techniques. The originality of our approach is to split customer data between the merchant and a semi-trusted third party, so that neither can derive sensitive information from their share alone. Therefore, the system can only be subverted by a coalition between these two parties. Experimental results confirm that the performance and user-friendliness of the application need not suffer from the adoption of such privacy-protection solutions. Furthermore, user testing of our prototype show that users react positively to the privacy model proposed.

1 Introduction

During the industrial revolution, judge Thomas Cooley defined privacy as being “the right to be let alone” [11] and later so did Warren and Brandeis [19] as “the right to left alone.” Since then, these definitions gradually evolved to

take into account other changes in the World, in particular those relating to new technologies. For instance, Westin defined privacy as the claim of individuals to determine what information about themselves is known to others, as well as when and how it is used [20]. That definition, which is still current, means that each individual must be the master of his person and of the data that he considers private.

In the context of e-commerce, it is important to know how to *classify* data as private or not. Indeed, the concept of privacy is relative in the sense that what is private for one person is not necessarily private for another [12]. In addition, it is important to investigate how to *preserve* privacy about individuals, while taking into account the classification that they made for their data. If classification can be taken care of relatively easily by the customer himself, privacy protection is far more difficult. In fact, apart from customers, privacy protection relies on the good faith of individuals and organizations that gather data on these customers in an open electronic environment such as the Internet.

In e-commerce, recommender systems allow entities providing goods or services to guide the choices made by potential *customers*. The recommendation can be issued by the merchant who sells the goods or services themselves or by intermediary brokers, but we shall henceforth use the generic term *merchant* to refer to the entity operating the recommender system.

One major drawback of recommender systems, however, is that in order to obtain accurate recommendations, cus-

tomers often have to reveal to the merchant information that they may consider private, such as their identity, demographic characteristics, previous buying behaviour, etc. It is important to keep in mind that potential privacy-preserving solutions to this problem should not be at the expense of the merchant's interests. In particular, the ability to make good recommendations might provide a competitive advantage that must be protected. Furthermore, the catalogue itself could have in some cases intrinsic commercial value. In both cases, the merchant should be able to protect its interests from competitors, and in particular from those who might be posing as potential customers and users of the recommender system. Thus, the problem of privacy protection in the use of recommender systems applies to both customers and merchants.

In earlier work [2, 3], we have proposed techniques to fight on behalf of the customer against profile creation or dossier constitution—and ultimately against the looming threat of Big Brother [10]. Our approach is designed to help the customer buy products without compromising the privacy of his profile. In particular, the merchant must be able to recommend products to the customer without knowing the latter's profile and identity. It may seem on first sight that this privacy concern goes against the philosophy of collaborative or demographic filtering (Section 2), in which each customer benefits from the preferences of other customers. Yet, we have shown that this is not necessarily so.

More precisely, cryptography and other security techniques can come to the rescue of privacy without sacrificing the genuine benefits that traditional recommender systems would offer. For this, we introduced in our previous paper a privacy-preserving hybrid recommender system, consisting of content-based filtering, demographic filtering and collaborative filtering. The main originality of our approach is the introduction of a *semi-trusted third party* that customers trust with certain private information but not with particulars of what they are looking for. In our system, the *Alambic agent* acts on behalf of this third party. In collaboration with the merchant, it provides to the customers accurate recommendations based on their private information. Our system is designed in such a way that neither the merchant nor the semi-trusted third party learn more about the customer than they should. The system can only be subverted by a collusion between the merchant and the semi-trusted third party.

Even though based on firm cryptographic foundations, our earlier paper can be seen as a proof of principle for our concepts because of its purely theoretical nature. The main purpose of *this* paper is to give a demonstration of feasibility for our ideas by reporting on their first experimental realization. For this purpose, we have implemented a small-scale web-based privacy-preserving recommender system. Our implementation comes with a questionnaire that allows users to provide us with valuable feedback at the end of

their experience. In this way, we have gathered (privacy-preserving!) data on 51 users, and we report on this as well.

After this introduction, we review basic concepts of recommender systems and privacy in Section 2. The next section describes the theory behind our system followed by its implementation. Section 5 describes the feedback provided by our initial users. We finish with our conclusions and future work in Section 6.

2 Recommender Systems and Privacy

Many approaches to recommender systems have been proposed in the literature. Among those described in [7], we concentrate on the following three techniques because we combine them through a *mixed* recommendation hybrid approach so that we can simultaneously offer recommendations based on all three of them: Collaborative Filtering (CF), Content-based Filtering (CN) and Demographic Filtering (DF).

Recommender systems are very useful, but they typically come at a price: in order to operate, it would seem that they need to collect information on those who use them. In the case of *collaborative* and *content-based* filtering, the customer typically provides the merchant with information on his buying behaviour, especially the products he has bought in the past, as well as his ratings on these products. The information given for the purpose of *demographic* filtering is even more sensitive. The potential abuses from unscrupulous merchants are obvious. They can use the data for many purposes that are illegitimate if not intended by the customer, such as data mining and the collection of statistics. Merchants could pool their information with other merchants and/or governments. They could also sell it. Information coming from diverse sources could be linked, resulting in the constitution of a formidable dossier on the customer [10]. All this could result in a serious erosion of the customer's privacy.

The problem of privacy-preserving recommender systems has been addressed before. In the case of Collaborative Filtering, this can be achieved by using *trusted identity proxies* such as *anonymizers* [6, 13]. The most serious (but not the only) drawback in this approach is that those proxies must indeed be trusted since the customer has to put his complete trust in one single entity. To get around this problem, a peer-to-peer privacy-preserving collaborative filtering scheme has been proposed [8, 9], which does not depend on the need to trust a proxy. However, this scheme requires several customers to be *on-line* simultaneously. Also based on peer-to-peer networks are the *personal recommenders* [15], which deliver recommendations on palmtop computers, in which personal data is either stored locally or shared in encrypted form. In other schemes [16], customers disturb their private data before sending them to

the merchant, using *Randomized Perturbation* techniques. More recent schemes [4,5] combine the use of a distributed community of users and data perturbation, in order to maintain the advantages of both techniques. In contrast to the approaches outlined above, we do not require several customers to be online simultaneously on a peer-to-peer network, nor do we introduce random perturbations into the data. Furthermore, no single party needs to be fully trusted.

3 Review of the ALAMBIC System

ALAMBIC is a generic architecture and protocol for building recommender systems employing simultaneously demographic, collaborative and content-based filtering, while attaining privacy objectives described below. This is achieved by combining well-known security primitives with the following principle:

“Trust no one, but you may trust two...”

The basis behind this *Division of Trust Principle* is that while customers might trust merchants with information about what they *want*, they do not trust them with information about who or what they *are* or what they *did*. Conversely, they might trust *someone else* with information about who and what they are, but not with information about what it is that they are looking for or what they previously bought. In recommender systems, this amounts to saying that customers will trust the merchant with the query on the database of available products, albeit a generic one, but with nothing else. On the other hand, they will trust another party with their demographic information, but they will not want it to know anything about what they bought or are now interested in, i.e. in what kind of transactions they have with merchants. Hence we refer to this additional actor as the *semi-trusted third party*.

The idea that such independent parties can be trusted is not a far-fetched notion: such is already the case for those bound by professional secret and other privacy laws. This includes, for example, the figure of a notary public or equivalently the attorney-at-law, but also, to a certain extent, governments and banks. Another obvious example, in this era of Internet and e-commerce, are the commercial Certification Authorities (such as *Entrust*, *Verisign*, etc.) whose business largely depends on the public's confidence in their honesty and professionalism. We believe that the use of the ALAMBIC system (or a system similar to it) would make it possible and practical to have such parties intervene in the e-commerce process to protect the privacy of the public.

In order to be able to compute recommendations, the Alambic agent distills an *essence*, which encodes sensitive information about the customers in a way that cannot be decrypted by any of the concerned parties alone, including the Alambic agent itself. Yet, the essence can

be used to produce recommendations by using standard cryptographic tools, including secure two-party computation (STPC) between the customer and the merchant.

Several studies have been conducted in order to categorize customers with respect to their privacy concerns [1, 12, 17, 18, etc.]. This has led us to identify four levels of privacy, which take account of the fact that different customers may have different privacy concerns.

1. **No privacy:** the customer does not care about the privacy of his personal information.
2. **Soft privacy:** the customer wants to keep his identity, demographic profile, purchase history and rating history secret from the merchant, but he does not mind if the merchant knows in which product(s) he is interested.
3. **Hard privacy:** the customer wants to keep his identity, demographic profile, purchase history and rating history secret from the merchant. Furthermore, he does not want the merchant to know in which product(s) he is interested nor anything about the recommendations he receives.
4. **Full privacy:** the customer wants to keep secret every component of his personal data. In particular, this includes his browsing behaviour as well as his actual purchase.

It is important to understand that the basic premise of our approach is that information that should be kept private from the merchant, such as purchase history and rating history, can be revealed without fear to the Alambic agent provided *it* cannot associate such information with specific products in the merchant's catalogue. Again, this Division of Trust Principle is essential for valid recommendations to be issued even though no single party has information about the customer that could be used to mount a dossier on him.

Another dimension in privacy, which is independent of the personal data listed above, concerns the *tracking* of customers by the entities involved. Even under soft, hard or full-privacy constraints, some users might not want the merchant to know that they are returning as the same person who shopped on a given previous occasion, even if anonymously [18]. With this in mind, we introduce the following terminology to account for the level of tracking that different customers might accept.

1. **Strong tracking:** The customer does not mind being tracked, even by the merchant.
2. **Weak tracking:** The customer does not want to be tracked by the merchant, but he does not mind being tracked by some third party such as the Alambic agent. This is particularly interesting if that third party does

not know what the customer is buying or even what appears in his request.

3. **No tracking:** The customer does not accept to be tracked under any circumstances by any parties. This constraint can be obeyed only if no entity, merchant or otherwise, is ever able to know whether a given user is in fact a previous user of the system. In particular, this means that the customer must provide anew whatever information is required to make recommendations each and every time he uses the system.

A crucial aspect of our approach is to allow each individual user to make a choice concerning both of these two independent components: privacy of personal data *per se* and tracking of users. In this way, we can cater to the priorities of diverse customers.

3.1 The ALAMBIC Architecture

The key components and concepts of the ALAMBIC architecture, described in detail in [2] and shown in Figure 1, are the following:

Merchant Platform. Offers recommendable services to customers. In particular, it hosts the merchant’s products catalogue, encrypted customer profiles and encrypted customer ratings.

Controller Unit. Provides communications between the Alambic agent and the customer, and potentially supervises them in order to detect possible collusions between them.

Alambic Agent. Computes recommendations for customers. As a semi-trusted party, it does not possess the ability to associate customers with the descriptions of products recommended to or preferred by them.

Still Maker Platform. Ensures proper deployment of the Alambic agent by the semi-trusted third party (the “still maker”) to the merchant’s platform. The platform generates a public key certificate unique for each Alambic agent.

Pseudonym. Identifies a customer through various sessions, which simplifies the construction and maintenance customer profiles by the Alambic agent.

Anonymous indexation. Anonymizes the product catalogue to prevent the Alambic agent from knowing the association to actual real products in the catalogue. The merchant creates the anonymous indexes from a random string of characters in such a way that the Alambic agent can not

Table 1. Anonymous indexation example

Anonymous index	Catalogue
005467	Movie description 1
003495	Movie description 2
...	...

infer any information about the catalogue from this information alone (Table 1).

In previous work, we have discussed that the ALAMBIC framework could be implemented in a variety of ways, depending on the needs and business model of the involved parties: the merchant and the still maker. More precisely, we have described both a *two-tier* and a *three-tier* architecture.

In a three-tier architecture, the client, the merchant and the Alambic agent reside on three separate platforms at different locations on the Internet. In a two-tier architecture, on the other hand, the Alambic agent resides entirely within the merchant’s platform. This last option has the advantage of best protecting the merchant, since the data (i.e. the essence) is completely within the control of the merchant: it can be backed up and access to it by external parties (including the still maker) can be restrained. It has, however, the important disadvantage that in order to protect the customer privacy, some form of *surrogate computing* (Section 4) must be implemented, where the Alambic agent’s code is protected against reverse engineering and tampering, and its internal data is kept inviolable and confidential.

3.2 ALAMBIC Protocol

We are now going to describe the intermediate steps involved in a two-tier implementation of the ALAMBIC system, by which a user approaches the system, makes a request, obtains a recommendation and provides feedback. The overall protocol is thus divided into the four separate steps: *initialization*, *filtering*, *recommendation*, and *feedback*. They are described below and are also shown in Figure 1.

Initialization. The customer receives the Alambic agent’s public key certificate from the merchant. He then verifies its authenticity through a third-party, such as a certification authority, and extracts the associate public key A . Using A , he ciphers his pseudonym ψ , demographic profile π and a secret τ which will serve to establish a session key between the Merchant and the Alambic agent.

Filtering. After deciphering the customer’s pseudonym and demographic profile, the Alambic agent proceeds to

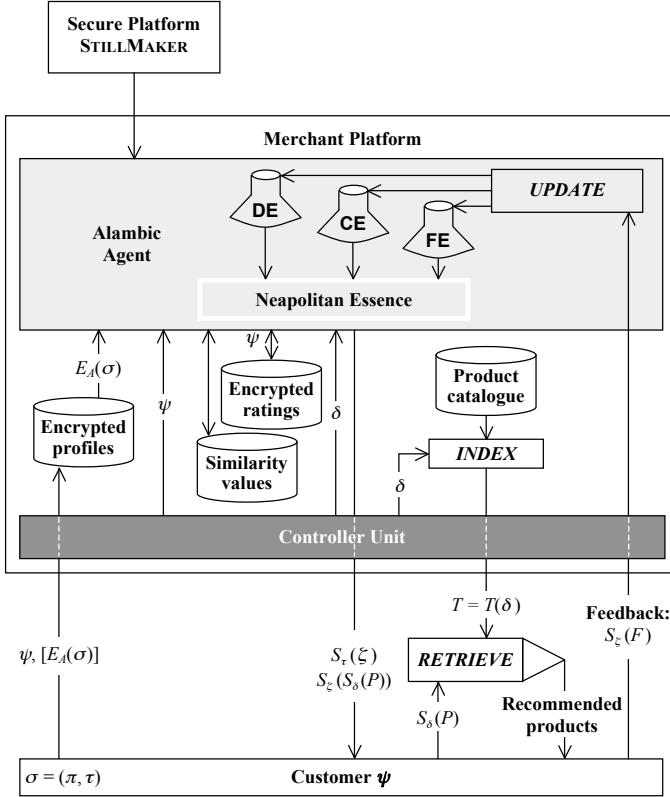


Figure 1. Architecture of the ALAMBIC system.

calculate recommendations using demographic, collaborative and content-based filterings. The resulting lists are merged using a hybrid filtering which produces a list P containing anonymous indexes.

Recommendation. Two secret session keys δ and ζ are generated by the Alambic agent to blind the combined anonymous indexes P , from the merchant and the customer. Key δ is applied first and serves to protect the indexes from customers who may exploit the map with the actual recommendations for competitive advantages. Key ζ , shared between the Alambic agent and the customer, protects the indexes from the merchant while they are being transferred. The Alambic agent safely sends ζ , by the use of the secret key τ initially obtained from the customer. The customer receives $S_\zeta(S_\delta(P))$ and can now execute the *RETRIEVE* procedure which allows him to receive the descriptions of recommendations without revealing to the merchant the anonymous indexes associated. Depending on the privacy requirements, this procedure can either be implemented by simply sending the deanonymized indexes to the merchant (soft pri-

vacy), or by using an STPC protocol that only reveals the descriptions to the customer and not to the merchant (hard privacy).

Feedback. The last step consists in sending a feedback to the Alambic agent. The feedback can be either explicit (e.g. rating items), or it can be an implicit feedback (e.g. purchasing items). The feedback provided forms a new list F which is composed of the anonymous indexes $S_\delta(p_i)$ and the associated ratings v_i . The list is then used by the Alambic agent to update the corresponding essences.

In summary, the *Initialization* step authenticates the Alambic agent to the customer, creates or authenticates a customer and also allows the Alambic agent and the customer to agree on a session secret key. The *Filtering* step deals with the filtering algorithms to produce a list of anonymous indexes. *Recommendation* converts the anonymous indexes into presentable recommendations that a customer can view. Finally, the *Feedback* step allows the customers to provide implicit or explicit feedback to the system which updates the essence accordingly.

In the absence of an actual implementation, our previous work on ALAMBIC left open two important questions. (1) Can these concepts *really* be deployed at a reasonable cost? and (2) What would real people think about it? The purpose of *this* paper is to report on the first experimental implementation of our approach, which we have made available for anyone to try out on the Internet.

4 The Alambic prototype

In our implementation of the ALAMBIC framework, we have chosen to implement a two-tier architecture, because it seemed to be *a priori* a more palatable option for merchants. In our previous work, we discussed several alternatives of how the integrity of the Alambic agent and the confidentiality of data contained within it could be both achieved, in particular *trusted computing hardware*, *code obfuscation and encryption*, and *sealed computing*.

Sealed computing simply refers to the capability of an isolated computer to store data and performing computing operations without allowing access to unauthorized parties. To protect the code and internal data, the computer can be placed within a hard-to-open container (e.g. a safe), upon which a tamper-proof seal can be posed. There are well-known and time-tested technologies that can be applied in this case, such that it would make it hard (though not impossible) to gain access to computer, but almost impossible to do so without being eventually detected.

In our implementation, and for simplicity, we chose to implement the two-tier architecture with a *surrogate sealed*

computer, as shown in Figure 2. The merchant’s platform consists of a single machine in which reside the front-end services visible to the client platform, i.e. the Web server and the controller that interacts with the client-side ALAMBIC-specific code. In addition, the back-end services, such as the database engine storing merchant data (such as the product catalogue) and the Alambic agent’s external data (the encrypted ratings and product similarity values), are also hosted on that machine. The surrogate computer contains only the Alambic agent’s code and internal data (program variables and the demographic essence) and sits within a separate private subnetwork, only accessible by the merchant’s front-end platform. The fact that this computer is “sealed” simply means that its contents cannot be accessed by the merchant without notice by a vigilant still maker. Other more tamper-proof alternatives would have been to use trusted hardware (such as a TPM) or to protect the code and data with obfuscation and encryption. In our case, the merchant’s platform has two network interface cards, one connected to the public network with a public IP address accessible from the Internet, and a second one connecting it to the private network containing the surrogate computer; the IP addresses on this subnetwork are private and are not reachable from the Internet.

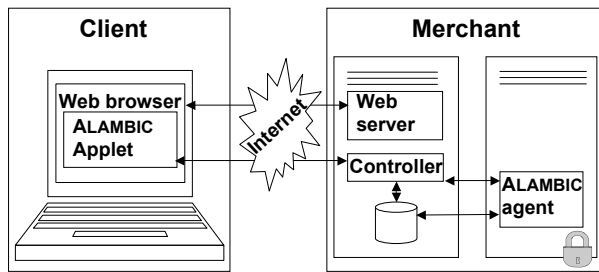


Figure 2. Architecture of the ALAMBIC prototype as implemented.

Note, however, that it is not absolutely necessary to host the database engine on the same machine as the Web server and controller. In most e-commerce applications, the Web server sits “exposed” within the network’s De-Militarized Zone (DMZ), while the database engine typically would reside on a separate machine that is not directly accessible from the Internet, deeper within the corporate network. Alternatively, the surrogate computer could run its own database engine, containing encrypted ratings and similarity values. This would be significantly more efficient, as it would obviate the need for the Alambic agent to use encryption and mask table access while getting and updating this data on the merchant’s database engine. The drawback would be that it would not be possible for the merchant to

protect the essence from a surrogate computer failure or accidental destruction. However, this risk can be reduced by implementing in the Alambic agent a backup feature that would upload the essences in encrypted form to the merchant’s platform on a regular basis. As pointed in [2], it might also be desirable for the Alambic agent to *reveal* the essence to the merchant, by providing it with a decrypted (yet anonymized) copy. If this is not done too often, it would allow the merchant to gain valuable insight on the global market trends and user preferences while protecting the privacy of users.

Finally, it is important to note one important limitation of the prototype. In this version, we chose *not* to implement an STPC protocol for the RETRIEVE procedure: the deanonymized indexes are simply sent by the customer to the merchant for a simple non-privacy preserving catalogue lookup which returns the product descriptions. Hence the prototype does not achieve hard privacy. The problem, as discussed in our previous work, is that on the one hand the user wants to obtain descriptions of recommendations based on anonymous indexes without revealing them to the merchant. On the other hand, the merchant does not wish to let the customer view his entire catalogue. While generic STPC protocols such as those introduced by Yao [21,22] theoretically can solve this problem, they are very hard to implement. Despite recent work on automatic protocol generators for STPC that can take “high-level” function descriptions as input [14], the practicability of such solutions in our context remains to be tested.

4.1 Prototype implementation

Beyond the standard software components (OS, Web server and database engine), the ALAMBIC prototype consists of three custom-made software components: the client-side component, the merchant’s controller unit and the Alambic agent. All of these custom-made components were programmed in Java, version 1.5.0 of the Java Runtime Environment (JRE). Communication between these components (Alambic agent and controller, client-side application and controller) are achieved by using standard socket programming methods: appropriate Java objects are passed back-and-forth at the various phases of the ALAMBIC protocol.

The client-side component of Alambic is a Java applet, running within the client machine’s Web browser. It was tested on various client OS platforms such as Windows, Linux and Macintosh, with the Internet Explorer, Firefox and Safari Web browsers. The applet code itself, which resides on the Merchant Web server, is distributed when the client connects. The applet is also signed using a public-key certificate, which would allow the client to verify that it is genuine code provided by the still maker. This ensures code

integrity in case the merchant or another party would want to present a rogue, non-privacy preserving applet. Code integrity allows us to hardcode Alambic agent's public key into the applet.

In our tests, the merchant platform was a dedicated desktop with a Core Duo 2.13 GHZ Intel processor and 2 GB of memory, running Gentoo distribution of Linux (kernel version 2.6). This machine also hosted the Web server (Apache version 2.0.58) and the database engine (MySQL distribution 5.0.26). The Web server's main function is to be the initial access point, from which the client-side Java applet is downloaded; it also hosts the system user manual and other static files. The database contains the product catalogue and the (encrypted) external data for the Alambic agent.

The controller unit is a standalone Java application running on the merchant's platform. The main function of the controller in our implementation is to relay information between the Alambic Agent and the client-side application; in addition it interacts with the database server to provide the client access to catalogue information. We purposely used port 443 (attributed to HTTPS) for the controller unit, as this port is normally supported by proxies and opened at most firewalls. Even though no SSL support was provided, this constitutes no breach of privacy since all sensitive information is encrypted by our custom applications. While this option provided maximum compatibility with the intended population of testers, another dedicated port should be used in production environments.

While the controller may also be used as a monitoring device to prevent the Alambic agent from leaking sensitive data, this functionality was not implemented. The merchant could normally create his own version of the controller, which would incorporate such monitoring functions. This could be achieved in a transparent manner to the rest of the system, since the client-side and Alambic agent API is well defined, and the port and protocol used to communicate with the database are standard.

The surrogate computer in our tests was also a dedicated desktop machine of the exact same characteristics (hardware and OS) as the merchant's platform. The only application running on it, however, is the Alambic agent, which is also a stand alone Java application. It communicates with both the controller unit and the database. Standard access control features of the database engine are used to prevent the Alambic agent from accessing anything but required information. A unique session identifier is used to track the current session between the Alambic agent and the client-side application; this is necessary since several clients might be using the system simultaneously.

4.2 Performance testing

Generally speaking, the performance overhead of ALAMBIC is comparable to that of other recommender systems. Beyond the operations already present in a typical recommender system, such as database manipulation and the computation of the actual recommendations, the only additional operation introduced because of privacy preservation that could have a significant impact on performance is the encryption/decryption of product catalogue indexes done by the various parties (client, merchant and Alambic agent). These operations, as we had predicted in earlier work, have negligible impact on performance. To verify this, we compiled two separate versions of the ALAMBIC software components: one with DES as the secret key encryption algorithm, and one in which no encryption was performed, or more precisely in which encryption/decryption is the identity function. With a limited-sized catalogue (15 items in our tests), the difference in performance between the two is negligible and hardly detectable by a human user.

However, in order to study the impact of catalogue size on these delays, we ran a benchmark on a test catalogue containing a variable number of anonymized indexes for fictional products. The total average encryption time by the merchant platform was 8.7 second for 1 million products, and 0.9 seconds for 100,000 entries. Note that this is the bottleneck, since the number of encryptions/decryptions by the client or the Alambic agent, would be significantly less; equal to the number of recommendations or the number of matching entries in a search query for the former, and the number of ratings for the latter.

Unfortunately, the initial setup of the client-side software takes considerable time. Because the client-side component is a Java applet, it must be downloaded every time from the merchant's Web server; this can take a few seconds. Worse yet is the startup time of the Java Virtual Machine, which can vary between 10 and 20 seconds, depending on the client machine's performance. Once this is done, typical transactions never take more than 1 to 3 seconds. Of course, there are several alternatives for reducing startup time, but these are all standard Web application engineering techniques, that are totally independent of the privacy protection features we have introduced here (and are outside of the scope of our research). For example, the use and distribution of a pre-compiled client-side application would greatly reduce initial startup and connection times.

In summary, the performance of the prototype is very good compared with typical recommender systems. Apart from an initial delay of several seconds, which could in principle be avoided in a production version, the cost of privacy-preservation is limited to 1 second for catalogue sizes up to 100 000 items, and 9 seconds for a 1-million

item catalogue. Nonetheless, commercial applications of ALAMBIC for catalogues of that size are likely to use high-performance hardware or hardware-based cryptographic accelerators, which could easily reduce that delay by an order of magnitude.

5 User Testing and Validation

We now describe the validation process of our prototype. There were 51 users who tested the system. When a new user enters ALAMBIC, he is presented with an overview of the system and an opportunity to learn more. He is then taken to a page in which he is invited to provide general information about his personal experience in online buying and his knowledge of recommender systems. Part of this general information is summarized in Table 2.

Table 2. Users previous experience.

Online purchasing		Recommender systems	
Never	5	Novice	25
Sometimes	37	Average	21
Often	9	Expert	5

The subsequent preliminary questions address directly privacy considerations. In our sample, 92% of users acknowledged that they were already aware of the gathering and the use of their personal information by online systems while interacting with them. Moreover, 59% (resp. 39%) of them were *concerned* (resp. *very concerned*) with their privacy while on the Internet. The final question at this stage concerned the introduction of our semi-trusted third party into recommender systems. First, we presented and explained the Division of Trust Principle. Then, the users were asked to indicate if they would trust an online system such as ALAMBIC, which is based on this principle. Overall, 80% of users said they would *sometimes* trust it, 12% *always*, and 8% *never*.

After answering those preliminary questions, the user is taken to the login page. See Figure 3. The user is first asked to choose among the four privacy and the three tracking levels. (An online *user manual* is always available in case the user has forgotten the meaning of those levels.) Depending on the tracking level, a login name and a password is requested from the user to allow the possibility of a repeat visit (strong or weak tracking level) or not (no tracking level). Users are also asked to provide the following demographic data: gender, age, income, and their highest completed education degree. At this point, they are ready to connect and get recommendations.

After they have finished using the system, users are taken to a final “System Evaluation” page, in which they

are asked to provide feedback. At first, they are asked to give an overall evaluation of the system with regards to the proposed levels of privacy: 59% found *good* the idea of introducing privacy levels, 29% *excellent*, 10% *average* and 2% did not express their opinion. The same question was asked about tracking levels: 53% found the idea *good*, 31% *excellent*, 12% *average*, 2% *poor* and 2% did not express their opinion. In addition, we traced the users according to their chosen level of privacy from the moment they started using the system. At the registration step, 67% of the users selected *soft privacy*, 20% *full privacy*, 10% *hard privacy* and 4% *no privacy*. Among those who selected soft privacy, some of them changed their mind after seeing the information the system gathers about them: 47% chose to keep the same level of privacy (soft privacy), 41% (resp. 12%) decided to move to hard privacy (resp. full) privacy. We noticed that even those who were not concerned by privacy at the beginning (4%) changed to soft and hard privacy after their experience. In contrast, people who chose hard and full privacy did not change their mind. What about the impact of the response time of ALAMBIC? 45% of users of the whole population decided to move to higher privacy levels (hard and full). Although the system informed them that the response time will increase, 74% maintained their decision. We also traced the users according to their chosen levels of tracking since they started using the system. Figures 4 and 5 compare the repartition of users at the registration step and the system evaluation step for privacy and tracking, respectively.

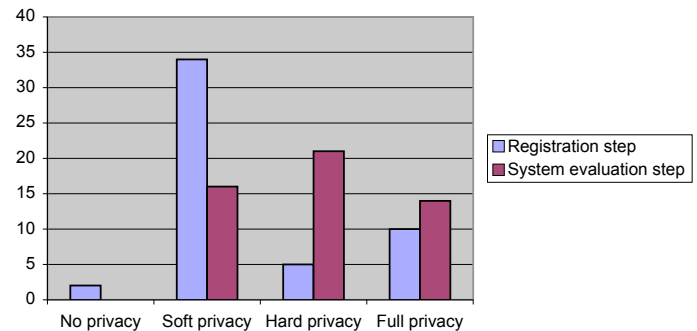


Figure 4. The evolution of privacy wishes among our users.

As a general remark, we can make the following statement: 92% of users acknowledged at the preliminary questions that they were aware about the gathering and the use of their personal information by online systems. After privacy levels being explained, there were 96% of users who in fact chose at least soft privacy.

Finally, users were asked to provide their overall satis-

Figure 3. Login interface.

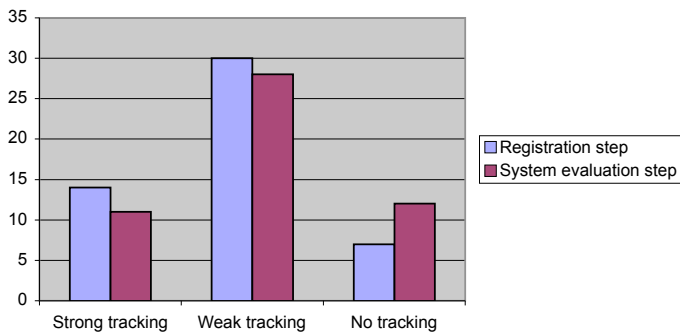


Figure 5. The evolution of tracking wishes among our users.

faction about the system: 65% found it good, 22% excellent and 13% average.

6 Conclusions

The prototype of the ALAMBIC framework that we have presented here has allowed us to accomplish two significant objectives. First, to verify the viability of implementing ALAMBIC, as a two-tier system using a sealed computer to implement the trusted computing paradigm. The performance overhead is acceptable and is due mostly to the initial setup time of the Java Virtual Machine (JVM), an engineering problem that we should be able to resolve in future versions of the system. As predicted, the overhead due purely to privacy protection features, such as encryption, is negligible for reasonable catalogue sizes (a hundred thousand or

less) and could most probably be addressed with adequate hardware investments for catalogues of several millions of items and more.

Secondly, the prototype has allowed us to conduct a battery of user validation tests in order to measure user responsiveness to such a system. These tests have clearly shown that users care about their privacy, even though the response time of the recommender system might increase slightly. In general, most users were satisfied by the privacy framework provided by ALAMBIC. It is also interesting to note that after interacting with the ALAMBIC prototype, users tend to change their mind toward higher levels of privacy and lower levels of tracking than those initially selected. There is also a strong correlation between the levels of privacy and the levels of tracking selected by users, indicating that both are similarly important to them. One recurring criticism received was that users were hesitant to provide their annual income, even to the trusted third-party, because this information was considered more sensitive than sex, age or level of education. However, this reticence might simply be a consequence of the fact that in this test environment, both the Alambic agent and the merchant platform were under the control of the same people: the authors of this paper.

In the near future, we plan on expanding the work to explore other solutions for the client-side component, in order to reduce long start-up times. We will also conduct further user tests. In particular, we will emphasise on the trust division principle validation by performing a pre-test and a post-test. In addition, an important aspect that should be tested is the potential recommendations improvement obtained from the trust gained by our system. Unfortunately, since in our tests we are at the same time the merchant and the Alambic agent, it is not appropriate to ask the users to

provide their real private information.

Lastly, and in order to be able to provide solutions for the full spectrum of privacy requirements, we are currently evaluating a possible replacement solution to the RETRIEVE procedure which will enable an efficient use of the system in hard and full privacy levels, without having to resort to a full STPC protocol implementation.

References

- [1] M. S. Ackerman, L. F. Cranor, and J. Reagle. Privacy in e-commerce: Examining user scenarios and privacy preferences. In *Proc. ACM Conf. on Electronic Commerce (CEC)*, pages 1–8, New York, USA, 1999.
- [2] E. Aïmeur, G. Brassard, J. M. Fernandez, and F. S. Mani Onana. ALAMBIC: A privacy-preserving recommender system for electronic commerce. *Int. J. of Information Security*. To appear.
- [3] E. Aïmeur, G. Brassard, J. M. Fernandez, and F. S. Mani Onana. Privacy-preserving demographic filtering. In *Proc. ACM Symposium on Applied Computing (SAC)*, pages 872–878, Dijon, France, 2006.
- [4] S. Berkovsky, Y. Etyani, T. Kuflik, and F. Ricci. Privacy-enhanced collaborative filtering. In *Proc. Workshop on Privacy-Enhanced Personalization (PEP)*, pages 75–83, Edinburgh, Scotland, 2005.
- [5] S. Berkovsky, Y. Etyani, T. Kuflik, and F. Ricci. Hierarchical neighborhood topology for privacy-enhanced collaborative filtering. In *Proc. Workshop on Privacy-Enhanced Personalization (PEP)*, pages 6–13, Montréal, Canada, 2006.
- [6] J. Boyan. The Anonymizer: Protecting user privacy on the Web. *Computer-Mediated Communication Magazine*, 4(9), 1997.
- [7] R. Burke. Hybrid recommender systems: Survey and experiments. *Customer Modeling and Customer-Adapted Interaction*, 4(12):331–370, 2002.
- [8] J. Canny. Collaborative filtering with privacy. In *Proc. IEEE Symposium on Security and Privacy*, pages 45–57, Oakland, CA, USA, 2002.
- [9] J. Canny. Collaborative filtering with privacy via factor analysis. In *Proc. ACM Int. Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 238–245, Tampere, Finland, August 2002.
- [10] D. Chaum. Security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044, 1985.
- [11] T. Cooley. *A Treatise on the Constitutional Limitations Which Rest Upon the Legislative Power of States of the American Union*. Callaghan & Co., Chicago, USA, 2nd edition, 1888.
- [12] S. Flinn and J. Lumsden. User perceptions of privacy and security on the Web. In *Proc. Conference on Privacy, Security and Trust (PST)*, pages 15–26, St. Andrews, New Brunswick, Canada, 2005.
- [13] E. Gabber, P. B. Gibbons, D. M. Kristol, Y. Matias, and A. J. Mayer. Consistent, yet anonymous, Web access with LPWA. *Communications of the ACM*, 42(2):42–47, 1999.
- [14] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella. Fairplay – A secure two-party computation system. In *Proc. of Usenix Security Symposium*, pages 9–13, San Diego, CA, USA, 2004.
- [15] B. N. Miller, J. A. Konstan, and J. Riedl. Pocketlens: Toward a personal recommender system. *ACM Trans. on Information Systems*, 22(3):437–476, 2004.
- [16] H. Polat and W. Du. Privacy-preserving collaborative filtering. *Int. J. of Electronic Commerce*, 9(4):9–35, 2005.
- [17] S. Spiekermann, J. Großklags, and B. Berendt. E-privacy in 2nd generation E-commerce: Privacy preferences versus actual behavior. In *Proc. ACM Conference on Electronic Commerce (CEC)*, pages 38–47, 2001.
- [18] M. Teltzrow and A. Kobsa. Impacts of user privacy preferences on personalized systems—A comparative study. In C. M. Karat, J. Blom, and J. Karat, editors, *Designing Personalized User Experiences for eCommerce*, volume 5 of *Human-Computer Interaction Series*, pages 315–332. Kluwer Academic Publishers, Dordrecht, Netherlands, 2004.
- [19] S. Warren and L. D. Brandeis. The right to privacy. *Harvard Law Review*, 4(5):193–220, 1890.
- [20] A. Westin. *Privacy and Freedom*. Atheneum, New York, 1967.
- [21] A. C.-C. Yao. Protocols for secure computation. In *Proc. IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 160–164, 1982.
- [22] A. C.-C. Yao. How to generate and exchange secrets. In *Proceedings of IEEE Symposium Foundations of Computer Science (FOCS)*, pages 162–167, 1986.