Reliable Medical Recommendation Systems with Patient Privacy

T. Ryan Hoens[1,2], Marina Blanton[1], Nitesh V. Chawla[1,2]
[1]Department of Computer Science and Engineering
[2]Interdisciplinary Center for Network Science and Applications (iCeNSA)
University of Notre Dame, Notre Dame, IN, USA
{thoens,mblanton,nchawla}@cse.nd.edu

ABSTRACT

One of the concerns patients have when confronted with a medical condition is which physician to trust. Any recommendation system that seeks to answer this question must ensure any sensitive medical information collected by the system is properly secured. In this paper we codify these privacy concerns in a privacy-friendly framework and present two architectures that realize it: the Secure Processing Architecture (SPA) and the Anonymous Contributions Architecture (ACA). In SPA, patients submit their ratings in a protected form without revealing any information about their data, and the computation of recommendations proceeds over the protected data using secure multi-party computation techniques. In ACA, patients submit their ratings in the clear, but no link between a submission and patient data can be made. We discuss various aspects of both architectures including techniques for ensuring reliability of computed recommendations and system performance, and provide their comparison.

Categories and Subject Descriptors

K.6.5 [**Management of Computing and Information Systems**]: Security and Protection; H.4.m [**Information Systems Applications**]: Miscellaneous

General Terms

Algorithms, Design, Reliability, Security.

Keywords

Recommendation systems, privacy, framework.

1. INTRODUCTION

It is evident that the health of an individual significantly affects her quality of life. For this reason, finding appropriate physicians to diagnose and treat medical conditions is one of the most important decisions that a patient must make. Currently, patients have two options that can aid them in addressing this problem, but both are of limited applicability. The first option is to rely on friends and family for advice on where to seek treatment. While recommendations produced by a close circle of friends can be assumed to be very trustworthy, the likelihood that friends and family have experience with the same medical history as the patient is quite low. Furthermore, such advice can often be unavailable when, for instance, a patient moves to a new area and does not have an established network from which to seek advice; even when this is not the case, the number of physicians which friends and family have had contact with may not adequately cover the options in the given area. The second option for patients is to seek public information about and/or ratings for a physician available on, e.g., the internet. Such ratings, however, are sparse as medical history is often treated as personal, confidential information. Public ratings also suffer from the problem of trustworthiness, as the likelihood of inaccuracies is higher.

To combat the problem of a paucity of experience among a patient's trusted friends and the limited value of the existing types of rating systems, we propose a framework which enables patients to gather reliable doctor recommendations for their condition(s) while protecting the privacy of both (i) the patients contributing their ratings to the system and (ii) the patients making inquiries. In this framework patients can rate physicians based on their satisfaction (defined on a per condition basis) affording the patients more fine-grained control over how to choose the physician who best suits their needs. It also protects the reliability of the results, meaning that (i) dishonest users cannot significantly influence the outcome of a physician's rating and (ii) no physician (or small group of users) has the ability to tamper with the ratings. This enables the system to maintain the integrity of its ratings and ensure they are as unbiased as possible.

As our privacy-friendly framework can be realized using a variety of techniques, we present alternative architectures. Because each alternative has its own advantages and disadvantages, we provide a fair and detailed assessment of the properties of each option, giving the community the ability to evaluate both. Moreover, certain options might be preferable over others in different contexts or application scenarios. Finally, we describe specific realizations of the architectures – which includes an implementation and experimental evaluation of the system – and report the results.

Our contributions can therefore be summarized as follows:

- development of a privacy-friendly framework for a reliable recommendation system of physicians using patient experience.
- presenting two architectures that realize the framework using different means from secure computation and anonymous communication techniques.
- design and implementation of specific protocols for the alternative architectures including experimental evaluation on a system prototype.

2. THE FRAMEWORK

In this section we develop the conceptual model of a privacy-friendly and reliable medical recommendation system by specifying its requirements and functional structure. These requirements will guarantee that patient privacy, as well as system and recommendation integrity, are maintained.

2.1 Functional Requirements

In our framework we assume that the system maintains a list of physicians and health conditions for which recommendations can be provided. Each contributing patient i submits her rating r_{ijk} for a specific physician j and specific health condition k. Each rating reflects the patient's satisfaction with physician j treating condition k, and is selected from a pre-defined and publicly known range. Without loss of generality, let this range be $[1, n]$, with the value of 0 be reserved for when no rating is available. The system will securely process or store the ratings to enable the following functionalities for any interested patient:

- A patient interested in health condition k should be able to obtain a physician recommendation for the condition based on the aggregated satisfaction information for all patients and all physicians treating the condition. Ideally, the recommendation is versatile enough to include alternative best-ranked physicians instead of providing only a single recommended name. That is, let s_{jk} denote the aggregate score for physician j on condition k computed from individual ratings r_{ijk}. (In this work, we use the term "rating" for individual values contributed by patients and the term "score" for the aggregate normalized value which is a function of individual contributions.) Then instead of learning the name of physician j with the highest score s_{jk}, the patient will be presented with a list of alternatives.
- A patient interested in a combination of health conditions $K = \{k_1, \ldots, k_\ell\}$ should be able to obtain a physician recommendation for the entire combination. Furthermore, the patient should be able to assign different weights $v_1, \ldots, v_\ell$ to the conditions based on their importance to the patient and obtain a recommendation that takes into account these weights. That is, the output will consist of best-ranked physicians where the ranks are determined using the weighted sum of the physicians' scores for the individual conditions, weighted by the patient-provided importance values v_i's, i.e., the combined scores are computed as $s_{jK} = \sum_{i=1}^{\ell} v_i s_{jk_i}$, where s_{jk_i} is the physician j's score for condition $k_i \in K$. As before, a set of alternatives is preferred over a single recommendation.

2.2 Rating Specification

When creating a recommendation system, an important consideration is providing users with accurate and relevant recommendations. In our system, we assume that the average rating given to a physician by her patients fits these criteria. Such ratings, however, can be the result of a wide variety of questions (e.g., overall satisfaction, time until cured, etc.), which are outside the scope of this work. When evaluating our system, we therefore assume that some overall rating for a physician exists. Moreover, we presume the rankings are numeric in nature, and have been normalized; this allows us to assume that an "average" rating makes sense, and is consistent across the recommendation system. Since none of the solutions presented in this work are reliant on any specific representation or source of scores, this does not affect any of the observations made in the paper.

In the rest of this work we assume that a recommendation is given for a specific condition (or a combination of conditions) and is computed from ratings submitted by patients. That is, patient i who has seen physician j for condition k can submit a rating r_{ijk} on the scale 1 to n. We use $r_{jk} = \sum_i r_{ijk}$ to denote the sum of all ratings for physician j on condition k and weight w_{jk} to denote the number of patients who contributed their ratings for physician j treating condition k.

2.3 Privacy Requirements

In the vast majority of existing recommendation systems, data contributed by users is assumed to be public information. In most cases this is a reasonable assumption, as user preferences are usually not sensitive in nature. While for most applications public ratings are acceptable, in medical applications they are not. This is due to the fact that even the existence of certain medical conditions is extremely sensitive data which the patient is highly unlikely to divulge. Therefore, expecting users to publicly disclose their opinion for doctors treating the conditions, without any assertion of privacy, is impractical. As such, we believe that recommendation systems which require users to divulge their recommendation (or even its existence) without provable privacy guarantees should be treated with suspicion.

This leads to the first privacy requirement of a recommendation system for medical applications: *the (lack of) existence of a recommendation for a patient is sensitive data which must be protected and unavailable throughout the lifetime of the recommendation system.* This also means that if at any point in time the patient's data is revealed to any entity, there should be no link between the patient's identifying information (e.g., name, IP address, etc.) and the data the patient contributed to the system.

Similarly, a user querying the system for a recommendation for a specific condition should not be forced to reveal that condition to the system. This means that the user will be able to obtain a recommendation for a specific condition or a combination of conditions without communicating her preferences to the system in unprotected form.

2.4 Reliability Requirements

In addition to patient security, physicians must also be protected from unreasonable users or dishonest competitors. That is, a small group of users should not be able to sabotage a physician's reputation. This is not limited to the patients who the physician treats, but also those who are refused, as well as other physicians competing for the same patients. Protecting against such users has the added benefit of creating a more robust recommendation system, thereby

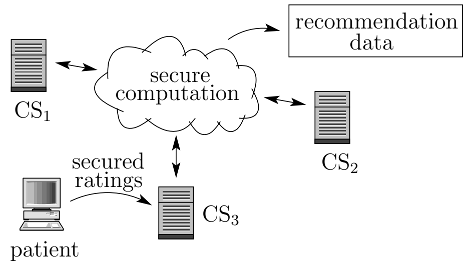

Figure 1: Submission of user ratings and recommendation computation in SPA architecture.

increasing its utility. Thus, the reliability requirement of medical recommendation systems can be stated as: *the reputation of the physicians in the system must be preserved, or at least the effectiveness of a small number of malicious users in altering physicians' scores must be mitigated*.

The above requirement immediately suggests two ways of dealing with dishonest users: malicious behavior can either (i) be prevented or (ii) detected and compensated for. As with any system that solicits input from a number of parties, each user can enter any rating even if it does not fully reflect her true experience. It is, however, possible to recognize several types of user misbehavior as abuse of the system. For example, a user can influence the aggregate score of a physician treating a certain condition by repeatedly submitting ratings for that physician. A user can also submit a rating which is out of the range (i.e., negative or above n) which has a larger effect on the physician's score than a single correct rating (since normally the aggregate score is computed as the average of individual ratings).

3. PROPOSED ARCHITECTURES AND REALIZATIONS THEREOF

Given the requirements presented above, we provide two broad classes of architectures which fit the framework. We call the first type the Secure Processing Architecture (SPA) and the second type the Anonymous Contributions Architecture (ACA). These architectures are described next, including their properties and engineering challenges associated with their realization. Concrete instantiations of the architectures, including implementation, are given in Sections 4 and 5, respectively.

3.1 Secure Processing Architecture (SPA)

In this architecture, as the name suggests, patients contribute secured (e.g., encrypted) ratings, and the computation of all recommendations is performed over secured data. The architecture, depicted in Figure 1, employs secure multi-party computation, where a number of computational servers collect data from patients and jointly compute recommendations. While identifying information of patients (who contribute or query data) may be available to the servers running the system, all submitted data is processed in a protected form and is not available to any entity. In the figure, computational servers CS$_i$ maintain the system and process patients' data. A contributing patient can properly secure her contribution and submit it to one or more computational servers. The servers engage in joint computation and make the recommendations available to queriers.

As customary in secure multi-party computation, a threshold scheme is used whereby the computation is performed by p computational servers with threshold t. In such schemes, any $t \leq p$ servers are able to successfully carry out or finish the computation, while any number less than t servers cannot learn anything about the data they handle. In this way the data remains secure assuming that t or more servers do not collude to learn any extra information. To maintain such security, in this framework the computational servers should be maintained by mutually distrustful or competing entities, so that any t of them are unlikely to conspire. For example, the computational servers can be run by (i) competing physician offices or hospitals, (ii) insurance companies, (iii) consumer rights protection organizations or programs, or (iv) a combination of the above.

In SPA, when a patient submits her secured rating for physician j and condition k, the computational servers should not be able to learn the value of j and k. The easiest way to hide this information is to have the patient submit a (secured) rating for each physician and each condition where only one submitted rating has the actual rating and carries a weight of 1, while all other submitted values have rating and weight 0. The servers will then be able to update the scores for all physicians and all conditions using the data received from the patient without the ability to know which particular value has been modified.

In particular, this can be accomplished as follows: the computational servers maintain (protected) sums r_{jk} and weights w_{jk} for each physician j and condition k. When a new rating r_{ijk} of weight w_{ijk} is submitted, the sum of ratings is updated as $r_{jk} + r_{ijk}$ and the weight is updated as $w_{jk} + w_{ijk}$. When r_{ijk} and w_{ijk} are both 0, nothing is modified. The scores s_{jk} can then be computed as the average rating r_{jk}/w_{jk} or any other function of r_{jk} and w_{jk}.

There are two common techniques for computing over protected data[1]: (i) encryption with special properties, called homomorphic encryption, which allows for operations on ciphertexts to translate into certain operations on the underlying plaintexts, and (ii) sharing the value to be protected among multiple parties and computing using its shares. Either technique will enable us to perform the computations outlined above, as well as all other computations necessary in computing recommendations. We chose to use homomorphic encryption in our instantiation of this architecture and its prototype implementation (Section 4).

Now notice that in this architecture the physicians' scores s_{jk} cannot be revealed because of the privacy requirements. That is, suppose that the computational servers post the scores which patients can use to compute necessary recommendations. Then when the next patient contributes her secured rating to the system and the scores get updated and published, it is likely trivial to find out what value, and for which physician and condition, the patient submitted a rating. Therefore, the aggregate scores must be protected as well, with only the recommendation data (such as a sorted list of best-ranked physicians) made available.

Because in this setup patients interested in learning recommendations have no impact on the way recommendation data is computed, there is a need to carefully design the function $f(r_{jk}, w_{jk})$ for computing scores so that it is useful and appeals to as broad of a population of users as possible.

[1]Other mechanisms exist as well, but are of limited applicability here.

We leave it to the community to determine what function is most meaningful for use in medical recommendation systems, but for the purposes of our realization, we propose to compute the scores as a combination of the average rating r_{jk}/w_{jk} and the number of patients treated w_{jk}. That is, set $s_{jk} = r_{jk}/w_{jk} + b_{jk}$, where $r_{jk}/w_{jk} \in [1, n]$, $b_{jk} \in [1, m]$, and n and m are chosen as desired. The purpose of b_{jk} is to let experienced physicians with a large number of patients have some advantage compared to physicians who treated a small number of patients for condition k. The value of m determines how much the extra factor b_{jk} influences the final score, and we propose to use a non-linear scale for the value of b_{jk}. Specifically, let $(t_1 = 0, t_2, \ldots, t_q)$ and $(b_1, \ldots, b_q)$, $q \leq m$, be two increasing sequences which will determine the value of b_{jk}. We set $b_{jk} = b_i$ (i.e., place w_{jk} in bucket i) if the value of w_{jk} is between thresholds t_i and t_{i+1}, i.e., $t_i \leq w_{jk} < t_{i+1}$ if t_{i+1} exists. The values of t_i's and b_i's can be set to any meaningful numbers as long as the sequences are increasing and $b_q \leq m$. Since the appropriate choice is not only disease dependent, but location dependent as well, we leave it up to the community to determine the bins. For example, we can have $n = 10$, $m = 5$, $b = (1, 2, 3, 4, 5)$, and $t = (0, 3, 5, 10, 20)$. This ensures that physicians who treated a sufficient number of patients will have the same value for b_{jk} (and thus the average ranking differentiates them), while physicians with a very limited number of visits will have a lower value of b_{jk}, and thus have to be rated more highly in order to rank ahead of their more experienced colleagues.

Given the above, a user who would like to learn a recommendation for condition k first obtains a list of the physicians sorted according to their scores s_{jk} (or a sorted list of top physicians only). We note that this outcome sufficiently hides individual contributions, where the physicians' ratings and the number of patients treated remain private. As secure multi-party techniques are relatively expensive, we suggest having the computational servers periodically compute the recommendations for all conditions and make that information publicly availably. In this way a patient interested in a specific condition is instantly able to obtain the desired recommendation. This also has the added benefit of hiding the recommendation of any individual, as the periodic update, if spaced appropriately, will include a large number of new contributions from many patients.

The system's design also allows patients to determine a custom rating based on a combination of conditions (where the combination is to remain private). In such cases, we first note that the number of diseases in a combination will be small since patients will seek separate specialists for unrelated conditions rather than one physician who can effectively treat all of them. Let u be the maximum number of conditions in commonly queried combinations (e.g., $u = 3$). Then the following options can be implemented within SPA: (i) the servers precompute and make available recommendations for each combination of $\leq u$ conditions for their choice of importance weights or (ii) the servers precompute recommendations for common combinations of $\leq u$ conditions and compute recommendations for other combinations upon user request (note that the results cannot be saved for any subsequent user to immediately obtain since the queried conditions are private). While the first option results in a higher load on the computational servers (as the number of all possible combinations grows rapidly, i.e., $O(n_c^u)$ for n_c conditions), the second requires patients with non-standard queries to experience delays. Also, combinations with non-standard weights will result in custom queries in both cases. In addition, while the choice of the conditions in a queried combination can be secured, the recommendation given to the querier (i.e., a sorted list of physicians) is likely to leak some information about the conditions. Thus, precomputed recommendations where a patient can retrieve information about all conditions at once is preferred from the patient privacy point of view.

In order to satisfy the reliability requirements of the framework, abuse of the system can be prevented using the following mechanisms. Each contributing patient can be required to submit only one rating r_{ijk} at a time. This allows the computational servers to detect an abnormal number of contributions from a particular user, treat the contributions as malicious, and disregard them. Additionally, when a patient submits a rating, she will have to prove that the submission is well-formed. This can be done through Zero-Knowledge Proofs of Knowledge (ZKPK), which prove the validity of certain statements over secured data without revealing any other information. In this application, the patient uses ZKPKs to prove that (i) all submitted pairs (r_{ijk}, w_{ijk}) except one are set to 0, (ii) there is weight w_{ijk} of value 1 and the corresponding rating r_{ijk} is in the range $[1, n]$. ZKPKs for all necessary functions such as OR, AND, equality, and a range are known and can be combined to prove the overall statement.

While designing a specific system using SPA, it is important to realize that there is a trade-off between privacy and computational overhead on one side and data reliability on the other. That is, instead of submitting $n_p \times n_c$ pairs (where n_p is the number of physicians and n_c is the number of conditions) to contribute a single recommendation, a patient might choose to submit fewer pairs for a subset of physicians and/or conditions. This will allow the computational servers to reduce the patient's conditions to a smaller set of potential diseases, but reduces the patient's computational overhead of sending a rating. This will also allow the computational servers to more effectively detect abuse of the system by dishonest users who repeatedly submit ratings for the same physician.

3.2 Anonymous Contributions Architecture (ACA)

In this architecture, unlike the prior approach, the patients submit their ratings in the clear. In order to ensure that there is no connection between a patient's identifying information and her contribution, all submissions are made anonymously. That is, patients use a system for anonymous routing to submit her contribution to the entity that receives all patient ratings and publishes information about them. Such anonymizer systems are readily available today (see, e.g., Tor anonymity network [3] and other anonymizer services and proxies such as [1, 2], among many others).

With this design, we can achieve the privacy requirements listed in Section 2. That is, the privacy of a patient who submits a rating r_{ijk} for physician j and condition k is not compromised because the recommendation system service learns no information about the user's identity and thus is unable to make any inferences about the health history of any particular patient. The service then processes all received ratings and makes aggregate information about the ratings available to all parties to use. This means that the privacy of all patients who would like to use the system is protected, as they can download the entire published table (i.e., information about all conditions) posted by the recommendation service and then disregard any irrelevant data. Alternatively, such users can use an anonymizer and retrieve only information about specific conditions from the published data.

To make the recommendation data available to the patients at least as flexible as in SPA, we suggest that the recommendation system service publishes the following information: for each physician j and condition k publish a pair $\langle r_{jk}/w_{jk}, b_{jk}\rangle$, where as before w_{jk} is the number of patients that contributed their rankings for physician j and condition k, $r_{jk}/w_{jk} = \sum_i r_{ijk}/w_{jk}$ is the average rating for physician j and condition k, and b_{jk} is the bucket value for w_{jk}. Availability of such data satisfies the functional requirements of Section 2. Furthermore, the data provides more information than the recommendations in SPA, as not only the ordering of physicians by their scores is known, but various other relevant information (e.g., the differences between the scores) can be computed as well. In particular, a patient can compute a recommendation for any combination of conditions using custom weights.

With respect to the reliability requirements, the anonymous nature of a patients' submissions can make the system prone to abuse. That is, dishonest users might attempt to influence a physician's overall rating by submitting bogus or repeated values. While it is trivial to defend against the former (i.e., all ratings are submitted in the clear, thus out-of-the-range or malformed values are immediately discarded as invalid), dealing with the latter requires architectural support. To combat this issue, we split the recommendation system service into two entities, called the Certification Authority (CA) and the Tabulating Authority (TA)[2]. The responsibility of the CA is to manage users, while the responsibility of the TA is to collect, process, and publish recommendation data.

The CA first registers users, issuing them anonymous credentials at the time of registration. Thus, users are required to register prior to submitting rankings to the TA. A registered user can then send her ranking to the TA and authenticate the submission using her anonymous credentials issued by the CA. We note that the authentication is anonymous in the sense that the only information revealed is that the user has been registered with the CA. By using a type of anonymous authentication (e.g., group signatures) which allows the identity of the sender to be uncovered under exceptional circumstances, abuse of the recommendation system service can be mitigated. That is, if the TA observes an unusually high volume of contributions that do not align with already stored data, a large number of submissions for the same (j, k) pair, or a number of extreme scores for the same physician that contradict the existing scores, the TA can provide the authentication records associated with the submissions to the CA who can then uncover the identity of the offending user(s). The process of when to request an identity to be uncovered can be controlled by a wide variety of parties including, but not limited to, the physicians.

Finally, because new users normally contribute for the first time shortly after their registration, we would like to prevent the CA from making correlations between the users that register and the content of the messages transmitted to

[2]As with multiple entities in SPA, it is recommended the CA and TA are run by different organizations.

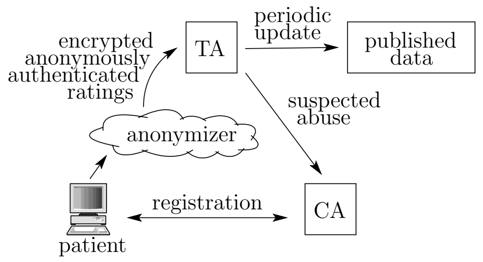

Figure 2: Submission of user ratings and recommendation computation in ACA architecture.

the TA over the network. For that reason, we assume that the TA has an encryption key and all contributions sent to the TA will be encrypted with that key. Furthermore, to prevent the CA from making similar correlations based on the updated information that the TA publishes, the updates by the TA to the published data should be infrequent, only after a significant number of new contributions (which will include contributions by both old and new users) has been accumulated. Note that this is not a major restriction of the system, as a small number of updates are unlikely to drastically affect the ratings of the physicians. The ACA architecture is depicted in Figure 2.

4. REALIZATION OF SECURE PROCESSING ARCHITECTURE

In this section we describe our particular realization of SPA. We first present the necessary background information, then define the full protocol, and conclude the description with a system implementation.

4.1 Preliminaries

As previously mentioned, techniques for implementing data protection in SPA include homomorphic encryption and secret sharing. In our solution, we use a semantically secure additively homomorphic public key threshold encryption scheme as a building block. The additive homomorphic property of such encryption schemes means that when one multiplies two encrypted messages, the result is a ciphertext that corresponds to an encryption of the sum of the messages. This property also implies that an encrypted message can be multiplied by a constant c by raising the ciphertext to power c. Semantic security means that no information about the underlying text can be learned from a ciphertext.

Recall that in a threshold public key encryption scheme anyone can encrypt using a public key, but decrypting values requires at least t out of p computational servers (for some $t \leq p$) to combine their keys to decrypt the value. In order to provide more security, we require the ability to generate the key material used in homomorphic encryption (i.e., the encryption and decryption keys) in a fully distributed manner. This requirement removes the need for a trusted party who has access to more data (i.e., the full key material) than anyone else. This is especially important in the current application, as this means that the security of each patient's data is not dependent upon any one party.

One scheme that encompasses the above properties is the Paillier cryptosystem [26], which we use in our implementation. That is, the Paillier cryptosystem is a semantically secure additively homomorphic encryption scheme, which can be used as a threshold scheme [18, 11] whose key generation can be performed in a fully distributed manner [4, 12].

In what follows, we use $[a]$ to denote an encrypted value of a. With the techniques we employ, some operations on encrypted values (such as addition and multiplication by a constant) can be performed locally, while other operations (such as multiplication or comparison of encrypted values) require interaction of the computational servers. Furthermore, comparison requires the operands to be available in bitwise form, which means that an encrypted value first needs to be transformed into an encryptions of its bits. We use $[a]_{\ell B}$ to denote encryption of the individual bits of a, i.e., $[a]_{\ell B} = \langle [a_0], \ldots, [a_{\ell-1}] \rangle$, where $a_i \in \{0, 1\}$ for $i = 0, \ldots, \ell-1$ and $a = \sum_{i=0}^{\ell-1} a_i 2^i$ (the length ℓ is explicit in the notation to permit a variable-length representation). It follows that our solution will need to rely on the following sub-protocols from the literature (see, e.g., [19, 28, 5]):

MULT: a protocol that, on input $[a]$ and $[b]$, produces $[a \cdot b]$.

BITS: a protocol that, on input $[a]$ and ℓ, produces encryption of ℓ least significant bits of the underlying plaintext of $[a]$, i.e., $[a]_{\ell B}$.

BIT-LE: a protocol that, given $[a_1]_{\ell B}$ and $[a_2]_{\ell B}$ outputs an encrypted bit $[b]$, where $b = 1$ iff $a_1 \leq a_2$.

4.2 Protocol

In the description of the protocol, for simplicity of presentation, we assume that physicians 1 through n_p are used to produce recommendations for condition k (in practice, some physicians with specialization far from condition k can be eliminated).

At a high level, the computation in the protocol proceeds as follows: First, for each physician we compute her score s_{jk} from the corresponding ranking (r_{jk}, w_{jk}). This means that the weight w_{jk} needs to be compared to all thresholds $t_1, \ldots, t_q$ as $t_i \overset{?}{\leq} w_{jk}$. After the results of the comparisons are available in the encrypted form as a binary vector of length q, we set the encryption of b_{jk} to the sum of computed bits in the vector, where bit i is weighted by the value $b_i - b_{i-1}$ (and by b_i when $i = 1$). Because the computed vector will always consist of a number of 1's followed by a number of 0's (if any), this approach computes the value of the bucket b_{jk} correctly. This mechanism minimizes the amount of interactive computation, and thus the amount of time, for the computation.

As a result of this step, we store s_{jk} as a numerator-denominator pair $([s'_{jk}], [w_{jk}]) = ([r_{jk} + b_{jk} w_{jk}], [w_{jk}])$. This representation allows us to finish the computation and sort the scores without performing expensive division operations. That is, to compare the scores of two physicians j_1 and j_2, we compare $s'_{j_1 k} w_{j_2 k}$ to $s'_{j_2 k} w_{j_1 k}$.

The above computation raises an interesting technical point in that the comparison must be performed correctly even if one or both of the scores have no contributions and are therefore 0. That is, if a physician j_1 has no ratings for condition k, both $r_{j_1 k}$ and $w_{j_1 k}$ are 0 (and thus $s'_{j_1 k} = 0$). According to the above comparison method, when the score of such a physician is being compared to the score of another physician who has a patient ratings, both values being compared will become 0, and the resulting outcome (as defined by the comparison protocol) is random. To ensure that the result of such comparisons is always correct, we modify the computation to add a flag which indicates whether w_{jk} is non-zero. That is, the comparison becomes $s'_{j_1 k} w_{j_2 k} + \text{non-zero}(w_{j_1 k}) \overset{?}{\leq} s'_{j_2 k} w_{j_1 k} + \text{non-zero}(w_{j_2 k})$. The output of non-zero$(w_{jk})$ is true (or 1) iff the OR of the bits of w_{jk} is 1. In the protocol, we denote this additional function by OR$([w_{jk}]_{\ell B})$, which will produce an encrypted bit. Notice that the function is not difficult to implement using a number of multiplications and additions.

An important observation is that this modification does not affect other comparisons in the system, i.e., when physicians j_1 and j_2 have ratings for condition k, $s'_{j_1 k} w_{j_2 k} + 1 \leq s'_{j_2 k} w_{j_1 k} + 1$ iff $s'_{j_1 k} w_{j_2 k} \leq s'_{j_2 k} w_{j_1 k}$. Similarly, when both physicians have no ratings, the result is unchanged.

In the protocol, we use various optimizations to ensure that its runtime is as low as possible. In particular, we use varying-length representation in bit decomposition BITS and comparison BIT-LE operations. Let ℓ_w denote the maximum length of counts w_{jk}, ℓ_s denote the maximum length of scores $s_{jk} = r_{jk}/w_{jk} + b_{jk}$ (i.e., if the ratings are in the range $[1, n]$ and the bucket values in the range $[1, m]$, $\ell_s = \lceil \log_2(n + m) \rceil$), and $\ell = 2\ell_w + \ell_s$ the maximum length of values used in the computation (i.e., for representation of $s'_{j_1 k} w_{j_2 k}$). We expect a normal choice of these parameters to be: $\ell = 32$, $\ell_w = 13$, and $\ell_s = 6$.

To optimize the performance of comparing weights w_{jk} to the thresholds t_i, notice that the weights w_{jk} can generally be longer than a value t_i, but by considering only $\log_2 \lceil t_i \rceil + 1$ bits we can always compare the values correctly. To form the $(\log_2 \lceil t_i \rceil + 1)$-bit representation of w_{jk}, we leave the $\log_2 \lceil t_i \rceil$ least significant bits of w_{jk} unchanged and replace the remaining bit with the OR of the remaining $\ell_w - \log_2 \lceil t_i \rceil$ most significant bits of w_{jk}. Thus, when the length of w_{jk} is greater than the length t_i, the optimized comparison will always result in w_{jk} being larger than t_i, otherwise the comparison proceeds as usual. When considering performance, note that such shortened representations of w_{jk} for the lengths of t_i's can be computed using $O(\ell_w)$ multiplications, regardless of the number of thresholds q.

The above optimization due to the variable-length representations allows us to reduce the runtime of the protocol by at least 40%. We are now ready to present the protocol.

```
RANK($k$, $([r_{1k}], [w_{1k}]), \ldots, ([r_{n_p k}], [w_{n_p k}])$):

  1. The computational servers set $\delta_1 = b_i$ and $\delta_i = b_i - b_{i-1}$; then for $j = 1, \ldots, n_p$ they compute in parallel:
     (a) execute $[w_{jk}]_{\ell_w B} \leftarrow$ BITS$([w_{jk}], \ell_w)$.
     (b) using $[w_{jk}]_{\ell_w B}$ compute shortened representations $[w_{jk}]_{(\log_2 \lceil t_i \rceil + 1)B}$ of $w_{jk}$ for $i = 2, \ldots, q$ and also compute $[f_{jk}] =$ OR$([w_{jk}]_{\ell_w B})$.
     (c) set $[c_1] = [1]$ and execute $[c_i] \leftarrow$ BIT-LE$(t_i, [w_{jk}]_{(\log_2 \lceil t_i \rceil + 1)B})$ for $i = 2, \ldots, q$.
     (d) locally compute $[b_{jk}] = [\sum_{i=1}^{m} c_i \delta_i] = \prod_{i=1}^{m} [c_i]^{\delta_i}$.
     (e) execute $[d] \leftarrow$ MULT$([b_{jk}], [w_{jk}])$ and locally set $[s'_{jk}] = [s_{jk} + d] = [s_{jk}] \cdot [d]$.
  2. The servers sort all tuples $([s'_{jk}], [w_{jk}], [f_{jk}])$ for $j = 1, \ldots, n_p$ using a suitable sorting algorithm and output the result, where each comparison is performed on $([s'_{xk}], [w_{xk}], [f_{xk}])$ and $([s'_{yk}], [w_{yk}], [f_{yk}])$ as follows:
     (a) execute $[v_x] \leftarrow$ MULT$([s'_{xk}], [w_{yk}])$ and $[v_y] \leftarrow$ MULT$([s'_{yk}], [w_{xk}])$.
     (b) locally compute $[v'_x] = [v_x] \cdot [f_{xk}] = [v_x + f_{xk}]$ and $[v'_y] = [v_y] \cdot [f_{yk}] = [v_y + f_{yk}]$.
```

(c) execute $[v'_x]_{\ell B} \leftarrow$ BITS$([v'_x], \ell)$ and $[v_y]_{\ell B} \leftarrow$ BITS$([v'_y], \ell)$.

(d) execute $[z] \leftarrow$ BIT-LE$([v_x]_{\ell B}, [v_y]_{\ell B})$ and open the value of z.

Note that in the above protocol the exact sorting algorithm is not defined. For our implementation we chose merge sort due to its simplicity, speed, and ease of being parallelized. The results are presented in the next section.

The security of the Rank protocol follows from the fact that only secure building blocks are used and the composition theorem [6] that states that composition of secure building blocks results in security of the overall construction.

To permit users to obtain recommendations for a combination of conditions K, we briefly discuss the modifications to the protocol above for two options: (i) the servers precompute the recommendation for conditions K using their choice of importance weights v_i's for the individual conditions in K and (ii) a patient asks the servers to compute a custom recommendation for her choice of conditions and corresponding weights which are to remain private. In the first case, the servers perform step 1 of the above protocol as specified for all physicians and all conditions in K. Then for each physician j with scores $([s'_{jk_i}], [w_{jk_i}])$ for $k_i \in K$, they compute the combined score $([s'_{jK}], [w_{jK}])$ as $w_{jK} = \prod_{k_i \in K} w_{jk_i}$ and $s'_{jK} = \sum_{k_i \in K}(v_i s'_{jk_i} \prod_{k_x \in K, i \neq x} w_{jk_x})$ in the encrypted form. Note that this computation uses only multiplications and additions, but results in values of larger length, which has an impact on the performance of sorting in the algorithm. Given such scores, step 2 of the protocol is then carried out unchanged using larger bit length.

To implement case (ii), the patient encrypts her weights v_i for conditions k_i in K. If the patient does not wish to reveal any information about the conditions in K, she can include all possible conditions and assign an importance weight of 0 to the irrelevant conditions. This will incur a significant overhead on the computational servers and unbearable wait time on the patient. To reduce the runtime, the patient instead can use only a few extra conditions in K to hide the ones in which she is interested. While this significantly reduces the computational overhead, this approach also leaks information about the patient's conditions of interest. (Note that the query result also may leak information.) The rest of the computation then proceeds as with option (i) with the exception that the weights v_i are processed encrypted.

4.3 Implementation

To test the feasibility of the protocol, we implemented it in Java using Paillier encryption with a 1024-bit key. Java was chosen due to its large number support and the ease of implementing distributed algorithms. All sub-protocols (i.e., BITS, MULT, BIT-LE) were implemented as in [19].

As the computations were distributed, we simulated each computational server as its own PC on a 100Mb LAN. The computers used were Dell workstations with 3.20 GHz Pentium 4 processors and 1 GB of memory. To make the tests more general, and applicable to a wider range of settings, we tested the computation separately in the two steps of the Rank protocol. By presenting timing results for each step individually we provide the ability to estimate performance of the protocol on a variety of different parameters (e.g., number of bits in each t_i, number of buckets, etc.).

To test step 1, we varied the length of bucket thresholds t_i used in determining the value of b_{jk}. Specifically, we timed step 1 using thresholds of size 2, 5, 10, 15, and 20 as measured in number of bits. Note that the higher threshold sizes on this list are present only to demonstrate how the techniques scale to larger values, as they are unlikely to be applicable to a recommendation system in practice. We also varied the number of physicians in the experiments; using 5, 10, 20, 50, 75, and 100 physicians. Given this, we provide a good estimate as to how long it will take to compute the scores for a specific number of physicians given an arbitrary number of buckets (with thresholds of varying sizes).

To test step 2, we timed how long it took to sort the various number of physicians when each was given a randomly assigned score and weight. The length $\ell = 32$ was used for bit decomposition and comparison. Combining this with the results obtained from the timing of step 1, we can then estimate how long it will take to compute the full protocol.

The data used when measuring the timing results was simulated. The use of simulated data does not adversely affect the timing results obtained, as the performance only depends upon: (i) the number and size of thresholds t and (ii) the performance of the sorting algorithm (as measured in the number of comparisons). The former is independent of the data and the sorting algorithm is not dependent on how the actual underlying data was obtained. To ensure that the simulated data accurately reflected a real world scenario, we generated a sorted list of unique ranking/weight pairs. We then permuted this list randomly, ensuring that the sorting algorithm had to perform the average number of comparisons to sort the list. This process was then repeated to determine the average time required to sort the list.

The results for step 1 are presented in Table 1. In this table the number of physicians is represented vertically, while the number of bits in threshold t_i is represented horizontally. To estimate the performance for thresholds t, one first chooses the row corresponding to the number of physicians (e.g., 5). One then chooses the columns which correspond to the number of bits in each t_i. Thus, if buckets are defined by thresholds 3, 16, 31, 100, 520, and 1000, the columns selected correspond to 2, 5, 5, 7, 10, and 10 bits, respectively. Note that this example is given for demonstration purposes only and is not meant as suggested values for t_i. Given these columns, the approximate timing is obtained by adding the denominator for the first threshold, and the numerator for the remaining thresholds. The denominator represents timing for running step 1 of the protocol using a single threshold of the specified size; and the numerator corresponds only to a single comparison with a threshold (of the specified size) in step 1(c) of the protocol. Thus in our example, we have $524 + 381 + 381 + 428 + 499 = 2212$ seconds (approximately 36 minutes) for step 1 (using the same thresholds for 100 physicians requires approximately 12 hours).

In practice, we suggest using thresholds $t = (0, 3, 5, 10, 20)$. While a practicing physician is likely to treat a significantly larger number of patients, the number of users who contribute to a recommendation system is likely to be significantly lower. Therefore, even a few patient ratings indicate significant experience in treating a particular condition. In this example, the largest threshold uses only 5 bits.

The performance of step 2 is presented by the lower curve in Figure 3. We see that the amount of time spent sorting 100 physicians is quite high (approximately 1 day). Note, however, that this computation needs to be done quite infrequently (e.g., semi-annually or quarterly), as the score for

Number of physicians	The length of bucket threshold value in bits					
	2	5	7	10	15	20
5	310 / 524	381 / 594	428 / 641	499 / 713	621 / 835	742 / 955
10	618 / 1047	765 / 1194	864 / 1294	1013 / 1442	1265 / 1693	1519 / 1948
20	1238 / 2094	1536 / 2393	1738 / 2594	2041 / 2897	2554 / 3410	3079 / 3934
30	1855 / 3139	2304 / 3588	2608 / 3892	3064 / 4348	3840 / 5124	4633 / 5917
40	2477 / 4191	3078 / 4793	3484 / 5199	4094 / 5809	5132 / 6846	6188 / 7901
50	3094 / 5237	3848 / 5991	4356 / 6499	5119 / 7263	6418 / 8561	7741 / 9884
60	3714 / 6287	4619 / 7192	5229 / 7802	6147 / 8720	7704 / 10277	9297 / 11870
70	4334 / 7335	5392 / 8392	6103 / 9103	7176 / 10176	8995 / 11996	10854 / 13854
80	4956 / 8385	6164 / 9594	6977 / 10407	8206 / 11635	10281 / 13710	12408 / 15838
90	5575 / 9433	6935 / 10793	7850 / 11708	9234 / 13092	11567 / 15425	13965 / 17823
100	6199 / 10488	7712 / 12001	8729 / 13018	10265 / 14554	12861 / 17150	15524 / 19813

Table 1: Timing results for step 1 of the RANK protocol. The denominator denotes the time required for computing the entire step 1 with a single threshold of specified size. The numerator denotes the time required to perform an additional threshold comparison of the specified size in step 1(c).

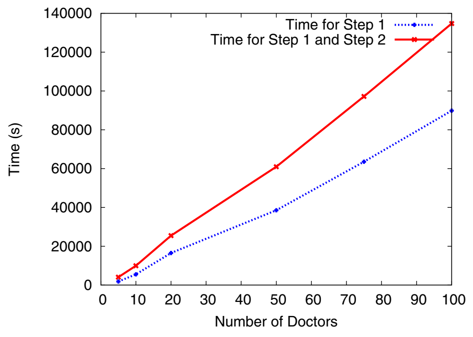

Figure 3: Timing results for steps 1 and 2 of the RANK protocol.

a doctor is not expected to be highly volatile. As previously mentioned, this also has the added benefit of providing more security for each of the users, as the wider the spacing between updates, the less likely that any one recommendation will be recognized.

The upper curve in Figure 3 shows the results of the overall RANK protocol. The difference corresponds to the runtime of step 1 using the 5 suggested threshold values. As we can see from this figure, the bucket computation time decreases in significance when compared to the amount of time used for sorting as the number of physicians grows. This is to be expected, as the sorting algorithm requires more comparisons than the bucket computation.

5. REALIZATION OF ANONYMOUS CONTRIBUTIONS ARCHITECTURE

Similarly to the previous section, here we describe our realization of ACA. We first present the background information, and then present a description of the protocol.

5.1 Preliminaries

In our description of ACA, we employ group signatures to implement the means of anonymous authentication that permits user identity to be uncovered in case of misuse. Group signature schemes, introduced in [9], allow any member of a given group to sign a message on behalf of the group. When any member of the group signs the message, her anonymity is preserved, i.e., no one inside or outside of the group can determine the signer of the message, only that an individual is in fact a member of the group. In such schemes, the entity who sets up the group possesses a private key which allows her to ascertain the identity of any message's signer.

In general, a group signature scheme consists of five algorithms: SETUP, JOIN, SIGN, VERIFY, and OPEN. The algorithm SETUP creates public parameters for the system and generates the master key msk. JOIN allows a user $\mathcal{U}$ to become a member of the group and obtain a key $gsk_{\mathcal{U}}$ by engaging in a protocol with the group manager. SIGN allows a member with a valid key to sign a message m anonymously on behalf of the group and produce signature $\mathsf{gsig}(m)$. VERIFY allows anyone with a group signature $\mathsf{gsig}(m)$ and the group's public key to verify the authenticity of a signature. OPEN is an algorithm that, given the master key msk and a signature $\mathsf{gsig}(m)$, allows the authority to recover the identity of the group member who signed the message. A secure group signature satisfies the security (i.e., message unforgeability by non-members) and privacy (i.e., inability to link two signed messages to the same user) properties.

Using this building block, the Tabulating Authority can be easily run by multiple entities who collect the data from the patients and post information about the combined data.

In our solution we also utilize Tor anonymizing network, which patients use to submit their contributions anonymously. In Tor [3], a message is routed through a number of participating hosts in such a way that each host knows only from what machine the message arrived and to which it should be sent next, but no other information (i.e., only the first Tor host will have information about the source and only the last one about the destination). This is achieved by multiple layers of encryption which are removed as the message moves through the network.

In ACA, the patients submit their information in an encrypted form. The easiest way to achieve this is to employ standard tools such as the widely used SSL/TLS suite [13, 14, 15]. Therefore, we assume that the TAs support the means of establishing secure channels via SSL which the contributing patients can use.

5.2 Protocol

Because the data processing in this architecture is over plaintext data (and the exact computation is as previously described), the most interesting component of ACA is data submission, which we detail next. Prior to any data submission, the CA runs the SETUP algorithm and announces the public parameters and keys pk of the system. Additionally, each entity serving the role of the TA publishes her public key which enables the patients to establish secure communication channels.

Register: User $\mathcal{U}$ and the CA engage in the Join protocol, during which the user obtains her credentials $gsk_{\mathcal{U}}$.

Submit:

1. Registered user $\mathcal{U}_i$ who would like to contribute ranking r_{ijk} first produces a signature $\mathsf{gsig}(r_{ijk}) \leftarrow \textsc{Sign}(r_{ijk}, gsk_{\mathcal{U}_i})$.
2. $\mathcal{U}_i$ uses Tor and SSL to establish a secure and anonymous connection with the TA.
3. $\mathcal{U}_i$ communicates the pair $(r_{ijk}, \mathsf{gsig}(r_{ijk}))$ to the TA through the established channel.
4. The TA verifies the signature as $r_{ijk} \overset{?}{=} \textsc{Verify}(\mathsf{gsig}(r_{ijk}), pk)$, and if the verification succeeds, sends the confirmation of the receipt through the channel.

Security and privacy of this interaction follows from the properties of the building blocks we use.

While the protocol for SPA required results from secure multi-party computation, this solution relies on group signatures and secure communication with anonymous routing. Since the amount of required computation is low, we do not provide the results of a sample implementation.

6. RELATED WORK

Prior research on privacy preserving recommendation systems has two main focuses: trust-based systems, and systems that preserve privacy of user preferences. In trust-based recommendation systems [20, 24], trust is modeled using a graph where each edge weight represents the trust between the two parties it connects. One example of the this is TrustWalker [20], where ratings of items are determined based on the trust network and an item similarity metric.

Work that focuses on preserving the privacy of user preferences is also prevalent. Unlike trust-based systems, these systems ensure that no user preferences are leaked during system operation. Such publications mainly focus on collaborative filtering (where a user has a set of preferences and the system attempts to suggest related products based on similarities with other users) which take user privacy into account. Most commonly the techniques are based on homomorphic encryption [7, 8] and data perturbation [27, 21, 29]. These techniques, however, were designed for a different problem than what we solve here.

An example more similar to our work is a system by Katzenbeisser and Petković [22]. In that system a secure medical recommendation is obtained by first encoding all relevant information (symptoms, diseases, etc.) into a standardized binary vector. The system then uses a matching protocol to determine which doctors have the best matching expertise via a secure matching algorithm, with the most suitable result returned as the recommendation. This solves a slightly different problem than our protocol, as we attempt to match patients with the optimal doctor for their relevant conditions, whereas their system makes no such guarantees.

In addition to privacy, another goal of our system is to be robust against misbehaving users. One common way misbehaving users attempt to influence the rating of a specific physician are known as "shilling attacks." Lam and Reidl [23] describe the attacks and discuss how they can affect the recommender system. Specifically, the authors consider various attack motivations (e.g., increasing/decreasing the rating of an item, hindering the credibility of the recommendation system as a whole) and their effect on recommendation systems. Importantly, they also note that while observing sharp changes in scores is an obvious way to detect (some) shilling attacks, non-trivial attacks against the system could potentially succeed. Detecting such attacks is proposed as a future area of research. Chirita, Nejdl, and Zamfir [10] provide further insight into shilling attacks and outline a detection algorithm which depends upon the distribution of scores that each user has made so far. The algorithm proved to be quite robust, providing not many false positives while catching many of the shilling attacks.

Another recent notion of privacy which is also used in recommendation systems is called "differential privacy." Loosely speaking, differential privacy [16, 17] for statistical databases guarantees that two data sets from the same population which differ by one element will occur with almost the same probability. Intuitively, this means that for each element in the database there exist very similar elements. Differential privacy is usually achieved by adding noise to the dataset, and the amount and type of noise is heavily dependent on the statistical queries that users are allowed to execute on the database. This means that there is a trade-off between accuracy and privacy. In application to recommendation systems, McSherry and Mironov [25] built a system to achieve differential privacy over the Netflix dataset. Experiments on their system showed that the accuracy of results was not severely effected throughout the query execution by maintaining privacy of user data.

7. CONCLUSIONS AND DISCUSSION

This work puts forward a framework for building medical recommendation systems in which (i) privacy of patient data is provably preserved, (ii) reliability of data is maintained by mitigating system abuse by dishonest users, and (iii) the functionality is flexible enough to provide recommendations on individual conditions as well as their combinations. We provide two alternative architectures, SPA and ACA, that satisfy the framework requirements.

While both architectures are functional and can be deployed in practice, each has drawbacks in comparison to the other. For example, in SPA:

1. Using modern secure multiparty techniques, computing recommendations incurs heavy computational load on the servers. The load depends on the number of physicians present in the recommendation system and the number of supported health conditions, but requires new recommendations to be produced infrequently.
2. Custom queries for user-specified combinations of conditions and weights cannot be executed often due to their computational cost on the servers (and significant wait time for users). Furthermore, such queries are likely to leak some information about the conditions included in the query (otherwise, the queries are infeasible to execute).
3. Honest users contributing to the system pay a (noticeable) computational price for proving correctness of their submission.
4. Finding mutually distrustful parties to perform the computations may not be easy (or possible); without such entities, the system does not guarantee privacy.

While in ACA we have:

1. Users must be willing to register with the system in order to make any contributions, but the concept of anonymous authentication might be difficult to explain to the average user. In addition, the fact that a user's identity may be revealed in exceptional circumstances may make a user believe that her privacy is not guaranteed, and thus hesitant to participate.
2. Privacy of user ratings is achieved by hiding among other users in the system. That is, when the system is not used by a large enough user base, privacy guarantees weaken. In the extreme case when there is only one registered user, no privacy can be achieved when a rating is submitted. The users of the system need to be aware of such potential vulnerability.
3. Dishonest behavior is more difficult to detect than in SPA. That is, in SPA the servers might decide to reject a contribution from a particular user if that user submits too many ratings (above a reasonable number of conditions that an individual or family might have). In ACA a large number of submissions by the same user is difficult to recognize and abuse needs to be detected based on other cues. A user wishing to change the rating of a particular physician might mask their behavior by submitting ratings for that physician with (fictitious) submissions for other physicians or conditions (thereby also potentially affecting ratings of other physicians). This behavior can go undetected unless the submission pattern significantly differs from the normal operation of the system. We, however, anticipate the abuse cases to be rare, especially since the users are made aware that there exists the ability to uncover their identity.

Based on this comparison, we leave it for the community to decide which architecture may be most beneficial for a particular context.

8. ACKNOWLEDGMENTS

This work partially supported by the NSF under grant BCS-0826958, the AFOSR under grant AFOSR-FA9550-09-1-0223, and the Richard and Peggy Notebaert Premier Fellowship.

9. REFERENCES

[1] Anonymizer, Inc. http://www.anonymizer.com.
[2] Shadowsurf. http://www.shadowsurf.com/.
[3] Tor: anonymity online. http://www.torproject.org.
[4] D. Boneh and M. K. Franklin. Efficient generation of shared RSA keys. In *CRYPTO'97*, pages 425–439, 1997.
[5] P. Bunn and R. Ostrovsky. Secure two-party k-means clustering. In *CCS'07*, pages 486–497, 2007.
[6] R. Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.
[7] J. F. Canny. Collaborative filtering with privacy. In *SSP'02*, pages 45–57, 2002.
[8] J. F. Canny. Collaborative filtering with privacy via factor analysis. In *SIGIR'02*, pages 238–245, 2002.
[9] D. Chaum and E. van Heyst. Group signatures. In *EUROCRYPT'91*, pages 257–265, 1991.
[10] P.-A. Chirita, W. Nejdl, and C. Zamfir. Preventing shilling attacks in online recommender systems. In *WIDM'05*, pages 67–74, 2005.
[11] I. Damgård and M. Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In *PKC'01*, pages 119–136, 2001.
[12] I. Damgård and M. Koprowski. Practical threshold RSA signatures without a trusted dealer. In *EUROCRYPT'01*, pages 152–165, 2001.
[13] T. Dierks and C. Allen. The TLS Protocol Version 1.0. RFC 2246 (Proposed Standard), Jan. 1999.
[14] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.1. RFC 4346 (Proposed Standard), Apr. 2006.
[15] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard), Aug. 2008.
[16] C. Dwork. Differential privacy. In *ICALP'06*, pages 1–12, 2006.
[17] C. Dwork. Differential privacy: A survey of results. In *TAMC'08*, pages 1–19, 2008.
[18] P.-A. Fouque, G. Poupard, and J. Stern. Sharing decryption in the context of voting or lotteries. In *FC'00*, pages 90–104, 2000.
[19] T. R. Hoens, M. Blanton, and N. Chawla. A private and reliable recommendation system using a social network. Technical Report 2010–05, Department of CSE, University of Notre Dame, 2010.
[20] M. Jamali and M. Ester. *TrustWalker*: a random walk model for combining trust-based and item-based recommendation. In *KDD'09*, pages 397–406, 2009.
[21] H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar. On the privacy preserving properties of random data perturbation techniques. In *ICDM'03*, pages 99–106, 2003.
[22] S. Katzenbeisser and M. Petkovic. Privacy-preserving recommendation systems for consumer healthcare services. In *ARES'08*, pages 889–895, 2008.
[23] S. K. Lam. and J. Riedl. Shilling recommender systems for fun and profit. In *WWW'04*, pages 393–402, 2004.
[24] P. Massa and P. Avesani. Trust-aware collaborative filtering for recommender systems. In *CoopIS'04*, pages 492–508, 2004.
[25] F. McSherry and I. Mironov. Differentially private recommender systems: Building privacy into the Netflix prize contenders. In *KDD'09*, pages 627–636, 2009.
[26] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT'99*, pages 223–238, 1999.
[27] H. Polat and W. Du. SVD-based collaborative filtering with privacy. In *SAC'05*, pages 791–795, 2005.
[28] B. Schoenmakers and P. Tuyls. Efficient binary conversion for Paillier encrypted values. In *EUROCRYPT'06*, pages 522–537, 2006.
[29] S. Zhang, J. Ford, and F. Makedon. A privacy-preserving collaborative filtering scheme with two-way communication. In *EC'06*, pages 316–323, 2006.