Reliable Medical Recommendation Systems with Patient Privacy

T. RYAN HOENS, MARINA BLANTON, AARON STEELE, and NITESH V. CHAWLA,
University of Notre Dame

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 that any sensitive medical information collected by the system is properly secured. In this article, 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 multiparty 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

Additional Key Words and Phrases: Recommendation systems, privacy, framework

ACM Reference Format:
Hoens, T. R., Blanton, M., Steele, A., and Chawla, N. V. 2013. Reliable medical recommendation systems with patient privacy. ACM Trans. Intell. Syst. Technol. 4, 4, Article 67 (September 2013), 31 pages.
DOI: http://dx.doi.org/10.1145/2508037.2508048

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

Part of this work appeared in *Proceedings of the ACM International Health Informatics Symposium (IHI'10)* [Hoens et al. 2010b].
This work was 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.
Authors' address: T. R. Hoens, M. Blanton, A. Steele, and N. V. Chawla, Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556; corresponding author's email: mblanton@nd.edu.

DOI: http://dx.doi.org/10.1145/2508037.2508048

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, for example, 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.

In order 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 can only minimally 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 two 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 and authentication techniques.
—Development of novel mechanisms for maintaining trustworthiness of the data in the presence of dishonest contributors.
—Design and implementation of specific protocols for the alternative architectures including experimental evaluation on a system prototype.

2. RELATED WORK

There has been considerable research into privacy-preserving recommendation systems. Originally, privacy was achieved in recommendation systems by giving user information to a trusted third party who then performs the necessary calculations with other trusted agents.

One problem with this early approach is that, in addition to privacy, in order to be useful, recommendation systems must be robust against misbehaving users. One common way misbehaving users may attempt to influence the rating of a specific physician is known as *shilling attacks*. Shilling attacks are said to occur when a user attempts to sabotage a competitor in order to make themselves look better. Lam and Reidl [2004] 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 and hindering the credibility of the recommendation system as a whole) and their effect on recommendation systems. Importantly, they note that while observing sharp changes in scores is an obvious way to detect (some) shilling attacks, nontrivial attacks against the system could potentially succeed. Detecting such attacks is proposed as a future area of research. Chirita et al. [2005] provide further insight into shilling attacks and outline a detection algorithm which depends on 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. In this work, on the other hand, instead of trying to detect system abuse, we concentrate on abuse prevention.

While in shilling attacks competing physicians attempt to sabotage each other, *bad mouthing* [Bankovic et al. 2011] is said to occur when a (potentially offended) patient attempts to lower the score of a physician. *Boosting* (or *ballot stuffing*) is said to occur if, instead of lowering a score, the patients collude to increase a rating [Dellarocas 2000; Srivatsa et al. 2005]. While we do not consider the implications of these attacks in this article, we note that the techniques in the literature to combat these attacks can be extended for our systems [Chirita et al. 2005; Burke et al. 2006; Mehta et al. 2007; Mobasher et al. 2006].

In addition to the problem of falsely modifying a score through illegitimate voting, another issue with the trust-based schemes is that they require users to trust the third parties to not misbehave with their data. In order to overcome this limitation, various techniques have been developed which do not require this third party. Two common ways of eliminating the need for the trusted third party are based on homomorphic encryption and data perturbation.

In the approaches based on homomorphic encryption [Canny 2002a, 2002b; Miller et al. 2004; Zhan et al. 2010; Berjani and Strufe 2011; Armknecht and Strufe 2011], users encrypt their data before sending it to any other party. This encrypted data is then used to compute the desired function without the underlying plaintexts being made available to any party. In this way, the user's privacy is preserved since no outside party is able to decrypt the ciphertexts (under certain constraints, e.g., when less than t participants collude). Note that these approaches based on homomorphic encryption all solve different problems than the one we solve here. That is, none of the approaches mentioned computing the weighted average; instead, they compute ratings based on similarity scores or some other function. Specifically, the method due to Canny [2002a, 2002b] addresses the problem of collaborative filtering which can be solved via expectation-maximization such that the update rules only require addition. PocketLens [Miller et al. 2004] is an alternative approach which is based on finding similar neighbors to a user who can, in turn, be used to compute good ratings for items. This is accomplished by computing the similarity of the ratings (via a dot product) of a user and various participants in the system. This similarity measure is then used to obtain accurate ratings for as-yet-unseen items. Zhan et al. [2010] propose a method for computing Pearson correlation. As the authors mention, the computation only requires multiplication and is therefore relatively easier than the one we describe in this article. Lastly, Armknecht and Strufe [2011] (building on the work of Berjani and Strufe [2011]) define a recommendation system based on Regularized Matrix Factorization (RMF). RMF was chosen because it provides accurate results and has many desirable properties: most notably, RMF can be learned via stochastic gradient descent. The ability to be learned by stochastic gradient descent means that the method can be trained without all instances in memory and is easily adaptable for use with homomorphic encryption.

An alternative to the homomorphic encryption-based approaches are the data perturbation approaches [Polat and Du 2005; Kargupta et al. 2003; Zhang et al. 2006]. In these approaches, users obfuscate their real data by adding noise to it before allowing it to be used in any computation. In this way, the users' actual ratings remain protected; yet due to aggregation, the authors argue that data perturbation effects still allow for effective recommendations to be made [Polat and Du 2005].

Another recent notion of privacy related to data perturbation approaches which has been used in recommendation systems is called *differential privacy.* Loosely speaking, differential privacy [Dwork 2006, 2008] for statistical databases guarantees that the outputs of a statistical query over two databases which differ by one element will differ by at most a negligible quantity. Intuitively, this means that the function is not significantly affected by any minor changes to the database. Differential privacy can be achieved by adding noise to the query, the amount and type of noise being heavily dependent on the statistical queries that users are allowed to execute on the database. This results in a trade-off between accuracy and privacy. In application to recommendation systems, McSherry and Mironov [2009] 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.

Specifically in the medical domain, one privacy-preserving recommendation system is due to Katzenbeisser and Petković [2008]. The authors developed a system in which 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 solutions, as we match patients with the optimal physicians for their relevant conditions, whereas the system of Katzenbeisser and Petkovic [2008] makes no such guarantees.

Finally, as we codified requirements for privacy-preserving medical recommendation systems, Chen and Williams [2010] codified requirements for privacy-aware social recommender systems. In their architecture, the authors argue that control, choice, and consent are the three main issues when developing a privacy-aware recommendation system. Taking these three issues into account, the authors then suggest adopting the privacy principles put forth by the Organization for Economic Cooperation and Development (OECD), adapting them to the recommendation domain.

It is important to note that our work differs from these publications in a few important ways. First, we describe a framework for medical recommendation systems which was designed to take the sensitivity of medical records into account. This includes the ACA and SPA architectures (Sections 6 and 5, respectively), which provide concrete realizations of the goals set forth in the framework. Second, whereas previous approaches obtain recommendations via collaborative filtering-based approaches, ACA and SPA allow for the computation of real-valued rankings of doctors (while still maintaining patient privacy), resulting in more fine-grained recommendations. These fine-grained recommendations also include the first instance of a description in the literature of a weighted average computation combined with physician's expertise using homomorphic encryption. Furthermore, while privacy-preserving recommendation systems based on secure computation (and homomorphic encryption in particular) exist, the anonymous contributions architecture is unique to this work.

3. 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.

3.1. Functional Requirements

In our framework, we assume that the system maintains a list of physicians and health conditions for which recommendations can be provided.[1] Each contributing patient i

[1] The list of physicians consists of all practicing physicians in one's area, and the conditions can be compiled according to the physicians' specialties from a standard set which can be expanded upon physician's or user's

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 predefined and publicly known range. Without loss of generality, let this range be $[1, n]$, with the value of 0 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, that is, 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.

3.2. 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 article.

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.

request. Standard ICD-9 codes can be used to uniquely represent conditions, which can be presented to the users of the system using common terminology.

3.3. 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 existence of a recommendation for a patient, or a lack of thereof, 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.

3.4. Reliability Requirements

In addition to patient privacy, 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 whom the physician treats, but also to 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 increasing its utility. Thus, the reliability requirement of medical recommendation systems can be stated as follows:

> 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.

This requirement immediately suggests two ways of dealing with dishonest users: malicious behavior can either (i) be prevented or (ii) be 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). We therefore target a system design that can effectively block these and similar types of system abuse.

4. PROPOSED ARCHITECTURES

Given the requirements just presented, 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

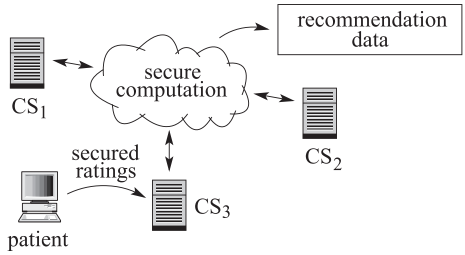

Fig. 1. Submission of user ratings and recommendation computation in SPA architecture.

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 5 and 6, respectively.

4.1. 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 multiparty 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 multiparty 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 these.

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 valid combination of physician j and condition k (i.e., for each physician j only the conditions that the physician treats are maintained). 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 overprotected data:[2] (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) splitting the value to be protected among multiple parties and computing using its shares. Either technique will enable us to perform the computations just outlined, 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 5).

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. 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 using nonlinear 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 $b_{jk} = b_i$ (that is, place w_{jk} in bucket i) if the value of w_{jk} is between thresholds t_i and t_{i+1}, that is, $t_i \leq w_{jk} < t_{i+1}$ 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 this, 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 multiparty 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

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

patients. For example, the rankings can be recomputed once a month or less frequently if the number of contributions since the last update is not sufficiently high.

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 nonstandard queries to experience delays. Also, combinations with nonstandard 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, and (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. We provide additional details in Section 5.3.

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 $\sum_{i=1}^{n_p} n_j$ pairs, where n_p is the number of physicians (who are numbered 1 through n_p) and n_j is the number of conditions physician j treats, to contribute a single recommendation, a patient might choose to submit fewer pairs for a subset of physicians and/or conditions. This might allow the computational servers to reduce the patient's physicians or conditions to a smaller set, 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.

4.2. 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 their contributions 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 Anonymizer, Inc.[4] and ShadowSurf[5], among many others).

With this design, we can achieve the privacy requirements listed in Section 3. 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 publish 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 3. 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 aid this issue, it can be beneficial to split the recommendation system service into two entities, called the Certification Authority (CA) and the Tabulating Authority (TA).[6] 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 ratings to the TA. A registered user can then send her rating 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 and is authorized to submit a contribution. In particular, the TA (or any other entity) is unable to tell whether any two contributions have been made by the same or different users.

By using a type of anonymous authentication which allows for enforcement of access control policies, abuse of the recommendation system service can be mitigated. That is, if at the time of submission the TA verifies that the (anonymous) user is authorized to submit a rating for physician j and condition k, the contribution will be accepted. For

[3] http://www.torproject.org.

[4] http://www.anonymizer.com.

[5] http://www.shadowsurf.com.

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

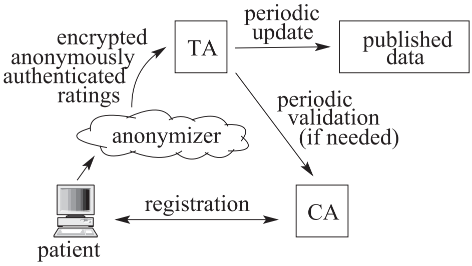

Fig. 2. Submission of user ratings and recommendation computation in ACA architecture.

this type of system, we consider two policies: (i) a bound on how many ratings a user is authorized to submit for a single (physician j, condition k) pair and (ii) a bound on the total number of submissions by a user. This will ensure that a single user cannot alter the aggregate score of a particular physician beyond a normal use and that a single user cannot have a significant impact on the system as a whole. We develop two realizations of ACA that support enforcement of both of these policies. The first is based on one-time-use credentials that the TA can update and reissue after each submission and which use new zero-knowledge proof techniques (detailed in Section 6.2) and the second is based on the novel use of electronic cash, which has a natural support for a distributed TA (detailed in Section 6.3).

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 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. Also, in the event that a malicious party conspires with a number of users or is able to obtain background knowledge about some of the ratings, the party will be able to reduce the set of observed changes in rankings between periodic updates to the remaining populace of the system.

The ACA architecture is depicted in Figure 2. A contributing patient interacts with the CA only during registration, and at that time, she is issued anonymous credentials. The rest of interaction occurs with the TA (which can be distributed and consist of multiple entities), and any querier would need to access only the published data. Depending on the realization of this architecture, there may or may not be direct periodic communication between the CA and the TA to validate credentials used in patients' submissions. Recall that the functional requirements of the system are met with this architecture since, given the published scores, a querier will be able to determine a ranked list of physicians for both individual conditions and a combination of conditions using weights of his choice.

5. 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.

5.1. 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 these properties is the Paillier cryptosystem [Paillier 1999], which we use in our implementation. The Paillier cryptosystem is a semantically secure additively homomorphic encryption scheme which can be used as a threshold scheme [Fouque et al. 2000; Damgård and Jurik 2001] whose key generation can be performed in a fully distributed manner [Boneh and Franklin 1997; Damgård and Koprowski 2001].

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 encryptions of its bits. We use $[a]_{\ell B}$ to denote encryption of the individual bits of a, that is, $[a]_{\ell B} = \langle [a_0], \ldots, [a_{\ell-1}] \rangle$, 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 subprotocols from the literature (see, e.g., [Hoens et al. 2010a; Schoenmakers and Tuyls 2006; Bunn and Ostrovsky 2007]).

—MULT. A protocol that, on input $[a]$ and $[b]$, produces $[a \cdot b]$. This is the simplest protocol that in our setting relies on additive splitting and additively homomorphic encryption. At a high level, the idea consists of splitting the (encrypted) second operand $[b]$ into random additive shares b_i such that $b = \sum_i b_i$ and making share b_i available to the corresponding server i in the clear. Multiplication is performed by each server on its own share using the homomorphic properties of the encryption scheme to obtain $[a \cdot b_i]$, after which the server assembles the product $[a \cdot b]$ in the encrypted form.

—BIT-LE. A protocol that, given $[a_1]_{\ell B}$ and $[a_2]_{\ell B}$, outputs an encrypted bit $[b]$, where $b = 1$ if and only if $a_1 \leq a_2$. For this functionality, we rely on the implementation described in Hoens et al. [2010a], in which the bit b is computed in the encrypted form from the (encrypted) bits of a_1 and a_2 by using a formula that consists of a number of additions, multiplications, and XOR operations. Since the XOR operation can be realized in terms of arithmetic operations as $x \oplus y = x + y - 2xy$, the overall complexity of this protocol is dominated by $O(\ell)$ multiplications MULT.

—BITS. A protocol that, on input $[a]$ and ℓ, produces encryption of ℓ least significant bits of the underlying plaintext of $[a]$, that is, $[a]_{\ell B}$. In this work, we utilize the protocol from Schoenmakers and Tuyls [2006], which has complexity proportional to ℓ. The protocol relies on the ability to multiply and invert encrypted values, generate random encrypted bits, and add and compare bit-decomposed values in encrypted form.

5.2. 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 any given condition k (in practice, some physicians with specializations 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 (aggregate) rating (r_{jk}, w_{jk}). This means that the weight w_{jk} needs to be compared to all thresholds $t_1, \dots, t_q$ 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 $s'_{j_1k}w_{j_2k}$ to $s'_{j_2k}w_{j_1k}$.

This 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_1k} and w_{j_1k} are 0 (and thus $s'_{j_1k} = 0$). According to the preceding 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 nonzero. That is, the comparison becomes $s'_{j_1k}w_{j_2k} + \text{nonzero}(w_{j_1k}) \overset{?}{\leq} s'_{j_2k}w_{j_1k} + \text{nonzero}(w_{j_2k})$. The output of nonzero($w_{jk}$) is true (or 1) if and only if 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, that is, when physicians j_1 and j_2 have ratings for condition k, $s'_{j_1k}w_{j_2k} + 1 \leq s'_{j_2k}w_{j_1k} + 1$ if and only if $s'_{j_1k}w_{j_2k} \leq s'_{j_2k}w_{j_1k}$. 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 $s'_{j_1k}w_{j_2k}$). 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 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 preceding 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 $\delta_1 = b_1$ and $\delta_i = b_i - b_{i-1}$. For $j = 1, \ldots, n_p$ they compute in parallel.
   (a) Execute $[w_{jk}]_{\ell_w B} \leftarrow \text{BITS}([w_{jk}], \ell_w)$.
   (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 $[f_{jk}] = \text{OR}([w_{jk}]_{\ell_w B})$.
   (c) Set $[c_1] = [1]$ and execute $[c_i] \leftarrow \text{BIT-LE}(t_i, [w_{jk}]_{(\log_2\lceil t_i \rceil+1)B})$ 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 $[d] \leftarrow \text{MULT}([b_{jk}], [w_{jk}])$ 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 $[v_x] \leftarrow \text{MULT}([s'_{xk}], [w_{yk}])$ and $[v_y] \leftarrow \text{MULT}([s'_{yk}], [w_{xk}])$.
   (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 $[v'_x]_{\ell B} \leftarrow \text{BITS}([v'_x], \ell)$ and $[v_y]_{\ell B} \leftarrow \text{BITS}([v'_y], \ell)$.
   (d) Execute $[z] \leftarrow \text{BIT-LE}([v_x]_{\ell B}, [v_y]_{\ell B})$ and open the value of z.

In this protocol, step 1 corresponds to computing ratings for each physician-condition pair. In particular, we first bit-decompose each weight w_{jk} in step 1(a), compute shortened representations of each bit-decomposed w_{jk} for performance reasons (by ORing the most significant bits which are being eliminated), as well as store the OR of all bits of w_{jk} in f_{jk} in step 1(b). The shortened representations allow us to faster compare (bit-decomposed) weights w_{jk} to the thresholds t_i in step 1(c), from which we compute the appropriate bucket values b_{jk} in step 1(d), and by the end of step 1 form ratings in the form $\langle s'_{jk}, w_{jk}, f_{jk} \rangle$, where $s'_{jk} = s_{jk} + b_{jk} w_{jk}$ and $f_{jk} = \text{nonzero}(w_{jk})$.

Step 2 sorts all rating tuples for each physician-condition pair, where a rating is first transformed into the form $s'_{jk} w_{jk} + \text{nonzero}(w_{jk}) = (r_{jk} + b_{jk} w_{jk}) w_{jk} + \text{nonzero}(w_{jk})$ during steps 2(a) and 2(b), bit decomposed in step 2(c), and compared to other ratings in step 2(d). The resulting ordering of ratings (i.e., physicians' rankings) is made public.

Note that in the preceding 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.

It is not difficult to show the security of the Rank protocol based on a standard definition of secure multiparty computation (see, e.g., [Goldreich 2004]). In particular, security in this context means that a protocol execution does not leak any information about the data being processed other than what can already be deduced by a participating party from its inputs and outputs alone, as intended. In our case, the security follows from the fact that only secure building blocks are used, and the composition theorem [Canetti 2000] that states that composition of secure building blocks results in security of the overall construction. Because the computational servers carrying out the protocol can be assumed to comply with their prescribed functionality, it is sufficient to use building blocks secure in the semi-honest model (in which the servers follow their prescribed computation, but might try to learn additional information about the data they handle). This would significantly improve performance of the solution compared to the setting where the computational parties are fully malicious (and thus can arbitrarily deviate from the protocol or disrupt the computation).

To permit users to obtain recommendations for a combination of conditions K, we briefly discuss the modifications to the preceding protocol for two options: (i) the servers pre-compute 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 preceding 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 protocol. 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.

5.3. Abuse Detection

Prior literature provides efficient zero-knowledge proofs of knowledge (ZKPK) for a variety of statements, with several efficient proofs for encrypted values and the Paillier encryption scheme in particular (see, e.g., [Damgård and Jurik 2001; Baudron et al. 2001; Cramer et al. 2001; Lipmaa et al. 2002]). Camenisch and Stadler [1997] introduced notation for various proof of knowledge, and we adopt their notation to our context. For example, we use

$$PK\{(\alpha, \beta) : A = [\alpha] \wedge B = [\beta] \wedge (\alpha = \beta)\}$$

to denote a ZKPK of knowledge and equality of the plaintexts corresponding to the ciphertexts A and B. The variables in the parentheses are not known to the verifier, but certain statements about them are being proved in zero knowledge. For the purposes of this work, we rely on the following ZKPKs from prior literature.

—*Proof of Plaintext Knowledge, Denoted* $PK\{(\alpha) : A = [\alpha]\}$. Such proofs have been developed in Damgård and Jurik [2001] and Baudron et al. [2001].

—*Proof that a Ciphertext Encrypts a Given Value, Denoted* $PK\{: A = [v]\}$. This proof can be executed as the preceding proof of plaintext knowledge followed by opening the plaintext value or by revealing the random coins used during ciphertext generation.

Table I. Comparison of ZK Shuffle Computation and Communication Cost for u Elements for Use with Paillier Encryption

Scheme	Prover's work	Verifier's work	Communication rounds	Communication
[Peng et al. 2005]	$7u$	$6u$	4	$12u\kappa_1$
[Groth and Ishai 2008]	$8u$	$7u$	3	$4u^{2/3}\kappa_1$
[Peng and Bao 2010]	$2u\kappa_2$	$u\kappa_2$	κ_2	$2u\kappa_1\kappa_2$
[Groth 2010]	$3u$	$2u$	4	$4u\kappa_1$

Note: Computation is measured in modular exponentiations and communication is measured in bits.

—*Proof that a Ciphertext Encrypts One Value from a Given Set, Denoted* $PK\{(\alpha) : A = [\alpha] \wedge \alpha \in S\}$. Such proofs can be found in or follow from the techniques of Damgård and Jurik [2001] and Baudron et al. [2001].

—*Proof of a Random Permutation (Shuffle) of a Set of Ciphertexts, Denoted* $PK\{(\pi) : B_1 = \pi(A_1), \ldots, B_u = \pi(A_u)\}$. During the shuffle, the ciphertexts are rerandomized so that it is not feasible to link a ciphertext from the initial set to a ciphertext in the permuted set. Such proofs have been developed (e.g., [Peng et al. 2005; Groth and Ishai 2008; Peng and Bao 2010; Groth 2010] among others). Groth and Ishai [2008] provide the first solution with a communication complexity that grows sublinearly with the number of ciphertexts in the shuffle. Peng and Bao [2010] provide the first solution to achieve perfect zero-knowledge. We provide a comparison of computation and communication costs of selected solutions from the literature for Paillier-encrypted values in Table I. The costs in the table are functions of the size u of the set being permuted, the work is measured in modular exponentiations, and the communication cost shows the dominating term. In the table, κ_1 denotes a security parameter that determines the ciphertext length (i.e., the RSA modulus length), and κ_2 is a correctness parameter for interactive proofs (where applicable), where the probability that the result is incorrect is $1/2^{\kappa_2}$. For Paillier encryption, each ciphertext size is $2\kappa_1$. While Groth [2010] introduces several shuffling schemes, we list in the table the scheme for shuffling known contents.

The overall statement that a user will need to prove in zero knowledge when submitting a rating r_{ijk} in our solution is as follows.

$$PK\{(\alpha, \pi) : A_1 = [\alpha] \wedge \alpha \in [1, n] \wedge A'_1 = [1] \wedge A_2 = [0] \wedge A'_2 = [0] \wedge \ldots \wedge A_u = [0] \wedge \\ \wedge A'_u = [0] \wedge (B_1, B'_1) = \pi(A_1, A'_1) \wedge \ldots \wedge (B_u, B'_u) = \pi(A_u, A'_u)\}.$$

Here, $u = \sum_{i=1}^{n_p} n_j$ and the ciphertext pairs $(B_1, B'_1), \ldots, (B_u, B'_u)$ are submitted as the user's rating. That is, the user prepares two sets of ciphertexts $(A_1, A'_1), \ldots, (A_u, A'_u)$ and $(B_1, B'_1), \ldots, (B_u, B'_u)$. She then shows that A_1 corresponds to an encryption of a value in the range $[1, n]$ (i.e., the user's rating r_{ijk}), that A'_1 encrypts 1 (i.e., the weight w_{ijk}), and that the remaining ciphertexts encrypt 0. The user then randomizes and permutes the ciphertexts (preserving each pair) so that the pair (A_1, A'_1) is placed in the location corresponding to physician j and condition k within the set of permuted (B_i, B'_i) ciphertexts and proves correctness of the shuffle. The servers use the permuted ciphertexts (B_i, B'_i) to update the aggregate ratings $([r_{jk}], [w_{jk}])$ for all physicians and conditions. Each user's contribution thus involves work (measured in cryptographic operations) linear in $u = \sum_{j=1}^{n_p} n_j$ with a low constant and can be substantially lowered if less stringent privacy guarantees are acceptable to the user.

5.4. Implementation

To test the feasibility of the RANK protocol, which is periodically executed by the computational servers, 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 subprotocols (i.e., BITS, MULT, BIT-LE) were implemented, as in Hoens et al. [2010a].

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 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 would 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 would 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 on (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. Instead the number of comparisons required is the most important factor when timing the protocol. To ensure that the simulated data accurately reflected a real world scenario, we generated a sorted list of unique rating/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 II. 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 1,000, 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

Table II. Timing Results for Step 1 of the RANK Protocol

Number of	The length of bucket threshold value in bits					
physicians	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

Note: 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).

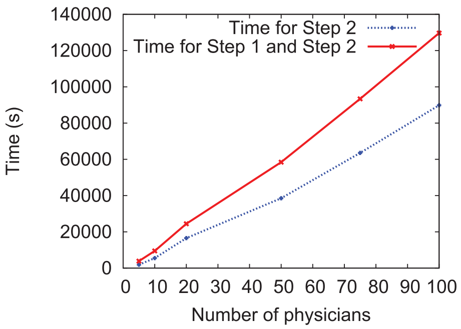

Fig. 3. Timing results for steps 1 and 2 of the RANK protocol.

(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 five 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 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 five 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.

Finally, we note that while Paillier encryption is well known and commonly used in secure multiparty computation protocols, recent progress shows that additively homomorphic encryption schemes can result in significantly faster performance. In particular, Damgård et al. [2008b, 2008a] provide an encryption scheme with shorter ciphertext size and faster encryption, decryption, and operations on ciphertexts. The scheme supports plaintext of small size and was originally developed for computing on encrypting bits. Blanton and Gasti [2011], however, used the scheme for a general-purpose computation (using the plaintext size ℓ comparable to the values used in this work) and showed a performance gain by about an order of magnitude on average compared to Paillier encryption. A threshold version of the scheme was not provided by the authors, but it shows the feasibility of significant computation speedup of this type of computation in the near future.

6. REALIZATION OF ANONYMOUS CONTRIBUTIONS ARCHITECTURE

Similar to the previous section, here we describe our realization of ACA. We first present the background information and then proceed with a description of two possible implementations of this architecture.

6.1. Preliminaries

6.1.1. Signature with Protocols. For this architecture, we make use of Camenisch-Lysyanskaya (CL) signatures, or signatures with protocols [Camenisch and Lysyanskaya 2002, 2004]. As customary for signature schemes, a signature scheme consists of key generation, signing, and verification algorithms, which we define at high level as follows.

—SETUP. On input a security parameter κ, creates and outputs a public-private key pair (pk, sk).
—SIGN. On input message m and private key sk, produces signature $\mathsf{sig}_{sk}(m)$.
—VERIFY. On input message m, public key pk, and signature $\mathsf{sig}_{sk'}(m')$ outputs 1 if and only if $\mathsf{sig}_{sk'}(m')$ is a valid signature on m under the corresponding private key sk.

What, however, distinguishes these CL schemes from ordinary signature schemes is that they come with additional protocols. Using the preceding schemes, it is possible to obtain a signature on a committed message m without disclosing it to the signer. In this case, a commitment to message m, denoted $\mathsf{com}(m)$, should be such that it hides the value of m (hiding property), but given $\mathsf{com}(m)$, it is infeasible to create another message $m' \neq m$ that matches the commitment $\mathsf{com}(m)$ (biding property). The Pedersen commitment scheme [Pedersen 1991] is commonly used with CL signatures for this purpose. We define the ability to sign committed messages by overloading the signing algorithm with an additional capability as follows.

—SIGN. On input commitment $\mathsf{com}(m)$ and private key sk, produces signature $\mathsf{sig}_{sk}(m)$.

To be more precise, in this type of commitment, information about m is unconditionally (or information-theoretically) protected, which means that that regardless of how many values an adversary tries, she will not be able to learn information about the message included in the commitment. This is achieved by using randomness that protects the value of m. For that reason, a commitment to message m can be expressed as $\mathsf{com}(m; r)$, where r explicitly defines the randomness used to form the commitment. Then, during signing, this randomness is used to form the signature, which more correctly should be defined as a signature on two messages, m and r, where as before, r protects m. More generally, a signature can be formed on any agreed upon number of messages $m_1, \ldots, m_k$, and we denote the corresponding signature by $\mathsf{sig}_{sk}(m_1, \ldots, m_k)$. Similarly, one can commit to several messages $m_1, \ldots, m_k$ in a single commitment, which we denote by $\mathsf{com}(m_1, \ldots, m_k; r)$. Then by invoking the signing algorithm, one can produce a signature $\mathsf{sig}_{sk}(m_1, \ldots, m_k, r)$ from that commitment. Because in our case, we always want the messages in a signature to be unconditionally protected (so that the signature can be consequently used without leaking any information about the values that it contains), we assume that the last message in a signature is always a random value and is not explicitly included in our notation. In other words, we, for instance, use notation $\mathsf{sig}_{sk}(m_1, m_2)$ to denote a signature on three messages m_1, m_2, and implicit r. Similarly, we omit randomness from commitment notation.

Using these signature schemes, it is also possible to prove knowledge of a signature on a message in zero knowledge (i.e., without disclosing any information about the message itself or revealing the original signature). We denote this functionality using the notation for ZKPKs as follows.

$$PK\{(\alpha) : A = \mathsf{sig}_{sk}(\alpha)\}.$$

Similarly, for a signature on multiple messages, we use notation

$$PK\{(\alpha_1, \ldots, \alpha_k) : A = \mathsf{sig}_{sk}(\alpha_1, \ldots, \alpha_k)\}.$$

This property is achieved by randomizing an existing signature in such a way that the randomization process does not invalidate its verifiability. This is necessary to ensure that each time a signature is used in a ZKPK, it will appear differently and cannot be linked to the past uses of the same signature in previous invocations of the same or different protocols. The preceding proof can be combined with other statements about the message(s) contained in the signature. For instance, the statement

$$PK\{(\alpha_1, \alpha_2) : A = \mathsf{sig}_{sk}(\alpha_1, \alpha_k) \ \wedge \ (\alpha_1 > 0) \wedge (\alpha_2 = 2\alpha_1)\}$$

proves two properties of the messages contained in a signature, namely, an inequality (or range) statement and a linear equation, solutions to both of which are known. In fact, we utilize both range proofs and proofs of linear relationships in our solution using the techniques of, for instance, Boudot [2000] and Camenisch and Stadler [1997], respectively.

In this context, the term anonymous credentials is used to refer to a signature issued by a certain authority on specific attributes that will allow the signature owner to anonymously prove certain statements about her attributes and obtain privileges based on such credentials.

In what follows, whenever the key is understood from the context, we omit it from the notation and, for instance, use $\mathsf{sig}(m)$ instead.

6.1.2. Other Tools. In our solution, we also utilize Tor anonymizing network,[3] which patients use to submit their contributions anonymously. In Tor, 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 [Dierks and Allen 1999; Dierks and Rescorla 2006, 2008]. Therefore, we assume that the TAs support the means of establishing secure channels via SSL which the contributing patients can use.

6.2. Solution Based on Anonymous Subscriptions

6.2.1. Description of the Technique. Blanton [2008] describes a mechanism for anonymous online subscriptions that relies on anonymous credentials and mitigates abuse of anonymity. While anonymous credentials based on CL signatures can be used to enforce proper access control (e.g., by verifying the type of subscription and the fact that a subscription's expiration date is in the future), they are prone to abuse by dishonest parties in that copies of the same subscription are valid credentials themselves which cannot be identified or linked to each other. Blanton [2008] proposed a mechanism in which chained one-time credentials are used for anonymous access. The idea is that at the registration time, a user obtains a CL signature on credentials as usual, but the signature also contains a randomly chosen message not known to the server. At the time of access, the user reveals that message to the server, who verifies that it has not been used before and verifies (in zero-knowledge) other credentials of the user. If the verification succeeds, the server grants the user access and reissues the anonymous credentials in the form of its signature on a new randomly chosen by the user message (not known to the server) and the remaining user credentials. With this design, there can be at most one chain of successful authentications to the server per user, which means that if the user duplicates her anonymous credentials and distributes them to other user, she denies herself access to the service. We build our first solution on this mechanism and propose novel zero-knowledge verification techniques that significantly improve the efficiency of the solution compared to what is readily available today.

As previously mentioned, the conditions that should be checked in our system include a bound on the number of times a participant can rank a specific physician for a specific condition and a bound on the overall number of contributions to the system by a participant. Because at the time of rating, a patient submits in the clear both the physician and the condition for which she is rating the physician, it is trivial to ensure that a patient does not rate a physician for a condition that that physician does not treat. Similarly, because the rating itself is submitted in the clear, it is easy to verify that the rating is indeed from the correct range $[1, n]$. We, however, would like to enforce the limit on the number of times an individual submits rating for physician j and condition k that the physician in question does treat, to ensure that the physician's score cannot be influenced by dishonest participants. For the same reason, we enforce the limit on the total number of contributions by a patient. We denote these two values for individual physician-condition and total contributions by c_{imax} and c_{tmax}, respectively.

Using anonymous credentials built from CL signatures, this can be achieved by issuing to a patient a signature on counts $c_{jk} = c_{imax}$ for each valid (physician j, condition k) pair and the count for the overall number of contributions $c = c_{tmax}$ at the registration time. Then during each submission, the TA learns the pair (j, k) and rating r_{ijk} that the (anonymous) patient intends to submit. Before accepting the rating, the TA verifies that indeed both counts c_{jk} and c in the user's credentials are greater than 0 (without learning any other information about them) and if the check succeeds, modifies the credentials by decrementing both c_{jk} and c by 1 (once again, without any knowledge of their values) and accepts the rating. Because each credential can be used only once, the patient will be forced to consequently use the modified credential, thus complying with the policy that limits both the number of contributions per physician-condition and the number of total contributions by an individual.

This solution, however, suffers from inefficiency. In particular, a participant's credential consist of a signature on $\sum_{i=1}^{n_p} n_j + 2$ messages, where n_j is the number of conditions that a physician treats. While a signature itself is compact and of constant size, proving that counts c_{jk} and c are above 0, however, involves work linear in the number of messages. In other words, the first step of proof of knowledge

$$PK\{(\alpha_1, \ldots, \alpha_u) : A = \mathsf{sig}(\alpha_1, \ldots, \alpha_u) \wedge (\alpha_i > 0) \wedge (\alpha_j > 0)\},$$

for known i and j, is to show that A is a valid signature on u messages, which requires u modular exponentiations for both the prover and the verifier. We, however, ideally would like to have a patient's work during submission of a single rating to be independent of the total number of physicians and conditions in the system.

To mitigate the problem, we propose packing several counts into a single message, which can significantly improve the efficiency of the necessary proofs of knowledge. To the best of our knowledge, this is the first time packing is used in signatures and zero knowledge protocols despite the popularity of such protocols. In detail, because each count can be represented using only a few bits and a signature can be issued on a message of significantly larger length, we combine several counts into a single message. Let ℓ_{ic} denote the number of bits used to represent a count for an individual physician and condition and ℓ_{tc} denote the number of bits used to represent a count for the total number of contributions. In our case, it is sufficient to set $\ell_{ic} = \lceil \log(c_{imax} + 1) \rceil$ (i.e., to be able to represent values between 0 and c_{imax}) and $\ell_{tc} = \lceil \log(c_{tmax} + 1) \rceil$, which will be justified later. Because the maximum number of contributions by a single user per physician and condition should be kept very low, we do not anticipate values greater than 2 to be used for ℓ_{ic}. Let q denote the maximum length of messages that can be signed using a CL signature scheme. For typical implementations, we expect q to range from 160 to 200. With our solution, we then pack $\lfloor \frac{q}{\ell_{ic}} \rfloor$ individual counts per message, which reduces the total number of messages included into a signature by a factor of 80–100. For simplicity, the count for the total number of contributions c is included as an individual message.

We obtain that at high level, the (ZK) proof that a patient will need to execute at the time of submitting her rating can be expressed as follows.

$$\begin{aligned} PK\big\{(\alpha_0, \alpha_1, \ldots, \alpha_u) \; : \; A = \mathsf{sig}(\alpha_0, \alpha_1, \ldots, \alpha_u) \wedge (\alpha_0 > 0) \wedge \\ \big(0 < \alpha_{i''} \bmod 2^{(i'+1)\ell_{ic}} - \alpha_{i''} \bmod 2^{i'\ell_{ic}} \leq c_{imax} 2^{i'\ell_{ic}}\big)\big\}, \end{aligned} \tag{1}$$

where $u = \sum_{j=1}^{n_p} n_j / \lfloor \frac{q}{\ell_{ic}} \rfloor + 2$ and i', i'' are known to both parties. We will consequently show how this functionality can be realized.

6.2.2. The Protocol. Prior to any data submission, the CA runs the SETUP algorithm of a CL signature scheme and announces the public parameters and key 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 with them. Public parameters of the system also include values c_{imax}, c_{tmax}, ℓ_{ic}, ℓ_{tc}, and the number of individual counts c_{jk} per signature message $n_m = \lfloor \frac{q}{\ell_{ic}} \rfloor$.

—REGISTER.

(1) User $\mathcal{U}$ computes $\mathsf{com}(m)$ for a randomly chosen value m of bitlength q and sends it to the CA.
(2) The CA produces additional messages to be included in $\mathcal{U}$'s credentials by setting $m_0 = c_{tmax}$ and $m_i = \sum_{j=0}^{n_m - 1} 2^{j\ell_{ic}} c_{imax}$ for $i = 1, \ldots, u$ (where m_u may contain fewer than n_m counts).
(3) The CA generates $\mathsf{sig}_{sk}(m, m_0, m_1, \ldots, m_u)$ using its sk and sends the signature to $\mathcal{U}$, who stores it as her anonymous credentials.

—SUBMIT.

(1) When user $\mathcal{U}$ would like to submit rating r_{ijk} for physician j and condition k, the user retrieves (or computes) index ind_{jk} for the pair (j, k) among the $\sum_{j=1}^{n_p} n_j$ physician-condition pairs. The user also computes $i' = \mathsf{ind}_{jk} \bmod \lfloor q/\ell_{ic} \rfloor$ and $i'' = \lceil \mathsf{ind}_{jk}/\lfloor q/\ell_{ic} \rfloor \rceil$.

(2) $\mathcal{U}$ prepares a commitment $\mathsf{com}(m')$ to a randomly chosen message m' of bitlength q.

(3) $\mathcal{U}$ randomizes her credentials $\mathsf{sig}_{sk}(m, m_0, m_1, \ldots, m_u)$ and sends them and $\mathsf{com}(m')$ to the TA.

(4) $\mathcal{U}$ and the TA engage in a ZKPK during which $\mathcal{U}$ opens the first message, m, in the credentials and proves in zero knowledge the following statement.

$$PK\big\{(\alpha_0, \alpha_1, \ldots, \alpha_u) \;:\; A = \mathsf{sig}_{sk}(m, \alpha_0, \ldots, \alpha_u) \wedge (\alpha_0 > 0) \wedge \\ \big(0 < \alpha_{i''} \bmod 2^{(i'+1)\ell_{ic}} - \alpha_{i''} \bmod 2^{i'\ell_{ic}} \leq c_{imax} 2^{i'\ell_{ic}}\big)\big\}. \quad (2)$$

(5) Upon successful verification of the ZKPK and the fact m has not been previously used, the TA records the rating r_{ijk} for physician j and k.

(6) The TA issues a new credential $\mathsf{sig}_{sk}(m', m_0', m_1, \ldots, m_{i''-1}, m_{i''}', m_{i''+1}, \ldots, m_u)$, where $m_0' = m_0 - 1$ and $m_{i''}' = m_{i''} - 2^{i'\ell_{ic}}$ are computed from the previous credentials without the knowledge of m_0 or $m_{i''}$.

Security and privacy of this interaction follows from the properties of the building blocks we use. In particular, the use of CL signatures and Pedersen commitments unconditionally hides information that could be used to identify the users and ensures that each interaction is unlinkable to prior interactions of that user with the TA. Furthermore, unforgeability of signatures and security of zero-knowledge proofs ensures that a user cannot bypass the verification process without valid credentials, and by our protocol design, duplication of user credentials is not possible.

While the protocol for SPA required results from secure multiparty computation, this solution relies on CL 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, that is, computing scores by the TA incurs minimal overhead (no cryptographic operations). Similarly, retrieving scores and computing recommendations by queriers does not involve cryptographic operations either. Submitting a rating involves the number of cryptographic operations on the order of u, which with our packing technique is not going to be high and can be performed in real time.

6.2.3. Proofs of Knowledge. In this section we focus on implementing a ZKPK of the form $(a < \alpha \bmod B_1 - \alpha \bmod B_2 \leq b)$ given a signed α, which is used in our protocol (and it is known how to implement the remaining portions of the overall zero-knowledge proof). First of all, this statement can be decomposed into the following expression, where each clause can utilize a single proof technique.

$$PK\{(\alpha, \beta, \gamma, \delta) : A = \mathsf{sig}(\alpha) \wedge (\beta = \alpha \bmod B_1) \wedge (\gamma = \alpha \bmod B_2) \wedge (\delta = \beta - \gamma) \wedge (a < \delta \leq b)\}.$$

Proof techniques for all of these statements are known (namely, a proof of knowledge of a signed value, a range proof, a proof of linear relationship of variables, and a conjunction of proofs), with the exception of a proof of equality modulo a certain value. The latter is therefore the focus of this section.

An expression of the form $\alpha = \beta \bmod B$ can further be decomposed into a statement of the form $(\beta = \gamma_1 B + \gamma_2) \wedge (\alpha = \gamma_2) \wedge (0 \leq \alpha < B)$ or $(\beta = \gamma_1 B + \gamma_2) \wedge (\alpha = \gamma_2) \wedge \alpha \in [0, B)$. The difference is that the former expression uses a range proof, while the latter uses a set membership proof which will be more efficient for small B. We, however, design a proof of knowledge of a statement $\alpha \equiv \beta \pmod{B}$, which has the cost of a single equality proof and therefore is more efficient than the preceding. Such a proof might also be of independent interest for other applications. With this type of proof, one can prove the statement $\alpha = \beta \bmod B$ as $(\alpha \equiv \beta \pmod{B}) \wedge (0 \leq \alpha < B)$.

Now we are ready to proceed with the proof that allows the prover to convince the verifier that $\alpha \equiv \beta \pmod{B}$ for unknown α and β in zero knowledge. As customary in the related literature, our solution is based on discrete logarithms. Informally, this means that the values about which we would like to prove a certain statement appear in the exponent. This is true for both commitment and signature schemes that we use. More formally, let G be a group of prime order p and g_1 and g_2 be generators of G. Then, for instance, with the Pedersen commitment scheme, commitment $\mathsf{com}(m)$ to $m \in \mathbb{Z}_p$ is formed as $\mathsf{com}(m) = (g_1)^m (g_2)^r$ for a random value $r \in \mathbb{Z}_p$ (note that all operations are in G, i.e., use modular arithmetic). The CL signature schemes that we use also have a similar form. For that reason, we next describe a proof in which on input $y_1 = (g_1)^{x_1}$ and $y_2 = (g_2)^{x_2}$, the prover shows that $x_1 \equiv x_2 \pmod{B}$ for $B < p$.

$$PK\big\{(\alpha, \beta) : y_1 = g_1^{\alpha} \wedge y_2 = g_2^{\beta} \wedge (\alpha \equiv \beta \pmod{B})\big\}. \tag{3}$$

It should be understood that we describe the proof in this form for simplicity of representation, and it achieves significantly weaker zero-knowledge guarantees than what our solution achieves. In particular, this proof is zero-knowledge in presence of a computationally bounded verifier and when x_1 and x_2 are large (i.e., the verifier cannot solve the discrete logarithm of y_1 to the base g_1 or of y_2 to the base g_2 and is also unable to brute force a significant portion of the overall space for α and β). A better option (which is used in our solution) is to use a signature or commitment where the values x_1 and x_2 are information theoretically protected, for example,

$$PK\big\{(\alpha, \beta, \gamma_1, \gamma_2) : y_1 = g_1^{\alpha} h_1^{\gamma_1} \wedge y_2 = g_2^{\beta} h_2^{\gamma_2} \wedge (\alpha \equiv \beta \pmod{B})\big\}.$$

In this case, the proof becomes zero-knowledge in presence of a computationally unbounded verifier. For simplicity of presentation we present a proof technique for statement in Equation (3), which can be compiled into this proof statement using known techniques for proving knowledge of a discrete logarithm representation (for y_1 and y_2 in this case).

We describe a standard type of a non-interactive proof which relies on a collision-resistant hash function $H() : \{0, 1\}^* \to \{0, 1\}^{\kappa_1}$, modeled as a random oracle for security purposes. Other types can be easily derived from our protocol. For security reasons, we require $|p| \geq \kappa_1 |x_i| + \kappa_2$ for a security parameter κ_2 that allows us to achieve statistical hiding (i.e., some information about the x_i's can be revealed with a probability near $1/2^{\kappa_2}$). Here, $|x_i|$ is the bit length of values used in such proofs, and the verifier is assumed to be honest.

(1) The prover generates two random values $v_1, v_2 \leftarrow \mathbb{Z}_p$ such that $v_1 \equiv v_2 \pmod{B}$ and computes $t_1 = g_1^{v_1}$, $t_2 = g_2^{v_2}$.
(2) The prover creates a commitment $c = H(g_1||g_2||y_1||y_2||t_1||t_2||B)$ (where "||" denotes concatenation), which serves the role of a challenge.
(3) The prover computes $r_1 = (v_1 - cx_1)$ and $r_2 = (v_2 - cx_2)$. If either r_1 or r_2 is negative, the prover restarts the proof from step 1. Otherwise, she sends $g_1, g_2, y_1, y_2, r_1, r_2, c$, B to the verifier.
(4) The verifier computes $t_1' = g_1^{r_1} y_1^c$, $t_2' = g_2^{r_2} y_2^c$, and $c' = H(g_1||g_2||y_1||y_2||t_1'||t_2'||B)$.
(5) If $c' = c$ and $r_1 \equiv r_2 \pmod{B}$, the verifier accepts the proof.

Note that the probability of failure in step 3 is negligible in the security parameter κ_2, which means that the prover will be able to successfully complete the proof on the first try with overwhelming probability.

We next need to show that this protocol is zero-knowledge proof of knowledge of the desired statement in presence of an honest verifier. To do so, we prove three standard properties sought of a ZKPK, namely, completeness (which states that an honest prover with possession of $g_1^{x_1}$ and $g_2^{x_2}$, where $x_1 \equiv x_2 \pmod{B}$ can indeed successfully complete the proof), soundness (which states that a prover with $x_1 \not\equiv x_2 \pmod{B}$ cannot successfully complete the proof with a nonnegligible probability), and zero-knowledge (which states that a simulator without access to x_1 and x_2 can produce a verifier's view indistinguishable from a real protocol execution).

Completeness. When $x_1 \equiv x_2 \pmod{B}$, the prover will be able to convince the verifier of this fact with probability 1. Since the prover's claim is true, the verifier will compute the following.

$$t_1' = g_1^{r_1} y_1^c = g_1^{(v_1 - cx_1)} g_1^{cx_1} = t_1 \text{ and } t_2' = g_2^{r_2} y_2^c = g_2^{(v_2 - cx_2)} g_2^{cx_2} = t_2.$$

This means that c must be identical to c'. Additionally, if $x_1 \equiv x_2 \pmod{B}$ and $v_1 \equiv v_2 \pmod{B}$, then it is straightforward to see that $r_1 \equiv r_2 \pmod{B}$. Therefore, the prover will be able to successfully complete the proof with probability 1.

Soundness. If x_1 is not congruent to x_2 mod B, to produce a forgery, the prover must generate x_1' and x_2' that will produce r_1' and r_2' that are congruent mod B. However, since the verifier will compute t_1', t_2' to be different than t_1, t_2 used by the prover, by our assumption that $H()$ is a random function the probability that $c' = c$ is $1/2^{\kappa_1}$. Therefore, the probability that a cheating prover can convince an honest verifier that $x_1 \equiv x_2 \pmod{B}$ is negligible in the security parameter κ_1.

Zero-Knowledge. We show that the verifier's view can be simulated without any knowledge of x_1 and x_2 and can be performed as follows.

(1) On input a random value c, the simulator chooses at random $v_1, v_2 \in \mathbb{Z}_p$ congruent modulo B and random x_1 and x_2 of the appropriate length also congruent modulo B.
(2) The simulator computes $r_1 = (v_1 - cx_1)$ and $r_2 = (v_2 - cx_2)$. If either value is negative, the simulator restarts from step 1.
(3) The simulator computes $t_1 = g_1^{v_1}$, $t_2 = g_2^{v_2}$, programs the random oracle for H to output c on input $(g_1||g_2||y_1||y_2||t_1||t_2||B)$, and outputs $g_1, g_2, y_1, y_2, r_1, r_2, c$, and B.

Now notice that under the assumption that the verifier is bounded (and the length of the x_i's is sufficiently large), the verifier's view is computationally indistinguishable from the protocol execution. The values of r_i's achieve stronger, statistical indistinguishability. Also, as mentioned earlier, if the proof is modified to work with commitments or signatures that perfectly hide the values of x_1 and x_2, the proof becomes statistical zero-knowledge in presence of a computationally unbounded verifier. This completes the security analysis of the proof.

The last part that remains is to show that the proofs of knowledge in Equations (1) and (2) do indeed enforce the necessary policy in our protocol. In particular, we want to show that a party with a signature on a count equal to 0 or less is unable to successfully pass the verification. In our case, this is necessary because the packed counts (and their negative values in particular) can interfere with each other within a single message.

With modular arithmetic, all values we operate on are positive, but if it is necessary to handle negative values, it is common to allocate half of the available space for negative values. For that reason, we might want to have the valid values $[1, c_{imax}]$ and $[1, c_{tmax}]$ for c_{jk} and c, respectively, occupy less than a half of the entire space (i.e., $2^{\ell_{ic}}$ and $2^{\ell_{tc}}$). Then checking for $(\alpha_0 > 0)$ can be accomplished by verifying that $(0 < \alpha_0 < 2^{\ell_{tc}-1})$. In this case, even if α_0 is decremented below 0, the proof will be successful only for users who are still authorized to contribute. In our implementation, however, the TA do not issue a renewed credential on a negative count (i.e., a credential is renewed and decremented only when the current count is above 0). This means that the lowest value that a credential can have is 0, which makes such cases even easier to handle in that support for negative counts is not needed. Similarly, we have that packed counts c_{jk}'s will not decrement below 0, but only negative values would affect other counts stored at lower bits within the same message. We thus obtain that ℓ_{ic} and ℓ_{tc} can safely be set to $\lceil \log(c_{imax}+1) \rceil$ and $\lceil \log(c_{tmax}+1) \rceil$, respectively.

6.3. Solution Based on Electronic Cash

Our second solution for implementing ACA consists of utilizing electronic cash (or e-cash) which can also be built from CL signatures [Camenisch and Lysyanskaya 2004]. E-cash schemes can vary widely in their implementations and properties, but in this work, we are interested in a particular type of e-cash that allows for offline spending of anonymous coins and can be built on CL signatures. Informally, in such schemes, a user obtains an electronic coin of denomination v by obtaining a signature from the bank $\mathsf{sig}(s, id, t, v)$, after which the bank withdraws amount v from the user's account. Here, s is the coin's serial number unknown to the bank (included into the signature using a commitment), id is the user's identity, and t is a random value. When the user would like to purchase goods from a merchant, she spends the coin anonymously as follows. The user opens the serial number s and denomination v and also releases the value $d = id \cdot R + t$ computed on the merchant's choice of R (the correctness of this value is accompanied by a zero-knowledge proof using a randomized version of the signature). At a later point, the merchant exchanges the coin it obtained from the user for monetary value v. To do so, the merchant submits s, v, R, d, and the proof of their correctness to the bank. If the value s has not been previously used, the bank pays amount v to the merchant. Otherwise, double spending took place. To determine whether the user double spent the coin (at a single or multiple merchants) or the merchant is trying to deposit the coin more than once, the bank retrieves the record it already has for s. If the current values d and R that the merchant is submitting differ from the previously stored values $\hat{d}$ and $\hat{R}$, the bank concludes that the user is at fault. In this case, the bank recovers the user's identity by solving a system of equations $id \cdot R + t = d$ and $id \cdot \hat{R} + t = \hat{d}$ for two unknowns id and t and applies a penalty to the user's account. Otherwise, if $R = \hat{R}$, the bank concludes that the coin was spent by the user only once, since it is the responsibility of the merchant to use unpredictable values of R. With such a solution, the user can anonymously spend coins offline. If the scheme is followed as prescribed, the user remains anonymous (i.e., the user's identity is information theoretically protected when the equation $id \cdot R + t$ is evaluated only on a single point R); if, however, the user double spends a coin, the anonymity is lost.

The preceding description demonstrates how two crucial features, namely, anonymous offline coin spending and double spending prevention, can be achieved using anonymous credentials. In our system, however, double spending prevention alone is not sufficient, and a mechanism for enforcing a substantially more complex access control policy need to be developed. The use of anonymous tokens that can be spent at a variety of entities, although, allows the TA to be realized in a distributed way and be run by multiple entities. In this section, therefore, we design a solution that utilizes electronic cash in a novel way to enforce nontrivial access control policy involving multiple conditions. To the best of our knowledge, this is the first time e-cash is used for this purpose and for enforcement of the conjunction of access control rules in particular.

At a high level, our solution consists of issuing different types of tokens, each of which can be used only once. A patient is issued c_{tmax} tokens that can be used to submit a rating for any physician and condition. For each physician j and condition k that that physician treats, the patient is also issued c_{imax} tokens that can be only used for rating physician j and condition k. Then, at the time the patient would like to submit her rating, the patient will be required to submit two tokens: one for the individual (j, k) pair and another general token that can be used toward any rating. This will allow the system to enforce the proper mechanism for mitigating system abuse.

One important consideration that must be taken into account when issuing multiple credentials to a single user is that of *collusion resilience*. In our context, collusion resilience means that tokens belonging to different users cannot be successfully combined to obtain access to resources which each of the users are not authorized to have individually. Suppose a certain user exhausts one type of tokens she is issued, for example, for a specific physician j and condition k pair, but has an abundance of tokens that can be spent on any physician. If the user obtains another token for the same physician j and condition k pair from another user (which the second user does not intend to use), she should not be able satisfy the system requirements by submitting two tokens, each of the correct type, that were issued to different users. We achieve collusion resilience by requiring that two tokens (of different types) submitted at the time of rating correspond to the same user, without revealing the identity of that user.

We next describe our solution in detail. In what follows, a (physician j, condition k) pair with the special value of (0, 0) will be interpreted as "any physician and any condition."

—Register.

(1) User $\mathcal{U}$ computes c_{tmax} commitments $\mathsf{com}(m_i, t_i)$ for randomly chosen values m_i and t_i and sends them to the CA.
(2) $\mathcal{U}$ also computes c_{imax} commitments $\mathsf{com}(m_i^{(jk)}, t_i^{(jk)})$ on random values for each valid (physician j, condition k) pair and sends them to the CA.
(3) Upon verifying $\mathcal{U}$'s identity id, the CA uses commitments $\mathsf{com}(m_i, t_i)$ and its secret key sk to issue c_{tmax} tokens $\mathsf{sig}_{sk}(m_i, t_i, id, \text{“}0||0\text{”})$ to $\mathcal{U}$.
(4) The CA also uses commitments $\mathsf{com}(m_i^{(jk)}, t_i^{(jk)})$ in conjunction with its key sk to issue c_{imax} tokens $\mathsf{sig}_{sk}(m_i^{(jk)}, t_i^{(jk)}, id, \text{“}j||k\text{”})$ to user $\mathcal{U}$ for each valid (j, k) pair.

—Submit.

(1) When user $\mathcal{U}$ would like to submit rating r_{ijk} for physician j and condition k, the user retrieves unused tokens of the type $\mathsf{sig}_{sk}(m', t', id, \text{“}0||0\text{”})$ and $\mathsf{sig}_{sk}(m'', t'', id, \text{“}j||k\text{”})$.
(2) The user randomized these signed tokens; let A and B denote their randomized versions. $\mathcal{U}$ sends to a TA of its choice values j, k, r_{ijk}, m', m'', A, and B.
(3) The TA provides $\mathcal{U}$ with a random challenge R and receives two values d_1 and d_2 as a response.
(4) $\mathcal{U}$ and the TA engage in a ZKPK interaction, during which $\mathcal{U}$ proves the following statement.

$$PK\{(\alpha_1, \beta_1, \alpha_2, \beta_2) \;:\; A = \mathsf{sig}_{sk}(m', \alpha_1, \beta_1, \text{“}0||0\text{”}) \wedge B = \mathsf{sig}_{sk}(m'', \alpha_2, \beta_2, \text{“}j||k\text{”}) \\ \wedge (\beta_1 = \beta_2) \wedge (d_1 = \beta_1 \cdot R + \alpha_1) \wedge (d_2 = \beta_2 \cdot R + \alpha_2)\}.$$

(5) Upon successful verification, the TA records all values received from $\mathcal{U}$ as well as a transcript of the proof, Φ, and accepts the rating r_{ijk}.

—Verify.

(1) When a TA would like to test the validity of a submitted rating r_{ijk} for physician j and condition k, the TA submits to the CA the values $j, k, m', m'', A, B, R, d_1, d_2, \Phi$ received from an (anonymous) user at the time of obtaining r_{ijk}.

(2) The CA searches its database of spent tokens for m' and m''. If neither of them is found, the CA verifies the correctness of the proof Φ against the values it received from the TA, records m', m'', R, d_1, d_2, and confirms the validity of the submission.
(3) Otherwise, if at least one of m' and m'' have previously been submitted, the CA investigates the case. Let $\hat{R}, \hat{d}_1$, denote values associated with a previously stored m'. The CA compares R to $\hat{R}$, and if they differ, recovers the *id* of the responsible user from R, d_1, $\hat{R}$, and $\hat{d}_1$. In this case, the CA notifies the TA of the system abuse, which causes the TA to discard the rating r_{ijk}. Otherwise, if $R = \hat{R}$, the CA notifies the TA of the duplicate verification request, which means that r_{ijk} was previously verified by the TA.
(4) If m'' was also found in the CA's database, the CA repeats step 3 for m'' using the respective values R, d_2, $\hat{R}$, and $\hat{d}_2$.

As before, security and privacy of our solution primarily follow from the properties of the building blocks that we use. That is, each contributing user remains anonymous as long as he complies with the agreed upon usage rules. All queriers also retrieve necessary scores and recommendation data anonymously. In addition, the rules for mitigating system abuse are enforced properly, as the user is unable to reuse her tokens, and collusion resilience is achieved through zero-knowledge proofs at the time of each contribution.

This solution also has a light computational overhead, as all ratings are received and processed in the clear. The storage of a contributing user consists of $c_{imax} \sum_{i=1}^{n_p} + c_{tmax}$ signatures and the same number of corresponding messages m_i's and $m_i^{(jk)}$'s. Because signatures are only tens of bytes long, this storage is not going to be a burden for the user. Finally, the work associated with each submission and verification protocol consists of a constant number of cryptographic operations and is thus very low. In particular, an implementation built on the signature scheme of Camenisch and Lysyanskaya [2004] would use bilinear maps with pairings implemented over elliptic curves. In such a setting, the equivalent of modular exponentiations is noticeably faster than in conventional cryptography, and the main overhead comes from performing a pairing operation (which for a typical set of parameters takes tens of milliseconds). The largest overhead of the solution then comes from signature verification that the server performs per contribution, while still allowing the server to perform all necessary computation within a second. The client's overhead is noticeably lighter, which gives a solution that can support a very large number of users.

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. Our realization of the SPA architecture relies on secure multiparty techniques which we enhance with misuse prevention techniques. Our realization of the ACA architecture relies on anonymous credentials and utilizes new zero-knowledge proof of knowledge techniques and a novel use of electronic cash for effective misuse prevention, as well as has lightweight computational overhead.

While both architectures meet all of the functional, privacy, and reliability requirements and can be deployed in practice, each has drawbacks in comparison to the other. For example, in SPA, the drawbacks are as follows.

(1) Using modern secure multiparty computation 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) Finding mutually distrustful parties to perform the computations may not be easy; without such entities, the system does not guarantee privacy.

While in ACA, we have the following drawbacks.

(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.
(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 (this affects SPA to a lesser degree because physicians' scores are not published). The users of the system need to be aware of such potential vulnerability.

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

REFERENCES

Armknecht, F. and Strufe, T. 2011. An efficient distributed privacy-preserving recommendation system. In *Proceedings of the IEEE Ad Hoc Networking Workshop*. 65–70.

Bankovic, Z., Vallejo, J., Fraga, D., and Moya, J. 2011. Detecting bad-mouthing attacks on reputation systems using self-organizing maps. In *Computational Intelligence in Security for Information Systems*. Lecture Notes in Computer Science, vol. 6694, Springer Verlag, Berlin Heidelberg, 9–16.

Baudron, O., Fouque, P.-A., Pointcheval, D., Stern, J., and Poupard, G. 2001. Practical multi-candidate election scheme. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*. 274–283.

Berjani, B. and Strufe, T. 2011. A recommendation system for spots in location-based online social networks. In *Proceedings of the EuroSys Workshop on Social Network Systems*.

Blanton, M. 2008. Online subscriptions with anonymous access. In *Proceedings of the ACM Symposium on Information, Computer and Communications Security (ASIACCS)*. 217–227.

Blanton, M. and Gasti, P. 2011. Secure and efficient protocols for iris and fingerprint identification. In *Proceedings of the European Symposium on Research in Computer Security (ESORICS)*. 190–209.

Boneh, D. and Franklin, M. 1997. Efficient generation of shared RSA keys. In *Advances in Cryptology—CRYPTO*. Lecture Notes in Computer Science, vol. 1294, Springer Verlag, Berlin Heidelberg, 425–439.

Boudot, F. 2000. Efficient proofs that a committed number lies in an interval. In *Advances in Cryptology—EUROCRYPT*. Lecture Notes in Computer Science, vol. 1807, Springer Verlag, Berlin Heidelberg, 431–444.

Bunn, P. and Ostrovsky, R. 2007. Secure two-party k-means clustering. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*. 486–497.

Burke, R., Mobasher, B., Williams, C., and Bhaumik, R. 2006. Classification features for attack detection in collaborative recommender systems. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 542–547.

Camenisch, J. and Lysyanskaya, A. 2002. A signature scheme with efficient protocols. In *Proceedings of the International Conference on Security in Communication Networks (SCN)*. Lecture Notes in Computer Science, vol. 2576, Springer Verlag, Berlin Heidelberg, 268–289.

CAMENISCH, J. AND LYSYANSKAYA, A. 2004. Signature schemes and anonymous credentials from bilinear maps. In *Advances in Cryptology–CRYPTO*. Lecture Notes in Computer Science, vol. 3152, Springer Verlag, Berlin Heidelberg, 56–72.

CAMENISCH, J. AND STADLER, M. 1997. Proof systems for general statements about discrete logarithms. Tech. rep. No. 260, ETH Zurich.

CANETTI, R. 2000. Security and composition of multiparty cryptographic protocols. *J. Cryptol. 13,* 1, 143–202.

CANNY, J. F. 2002a. Collaborative filtering with privacy. In *Proceedings of the IEEE Symposium on Security and Privacy*. 45–57.

CANNY, J. F. 2002b. Collaborative filtering with privacy via factor analysis. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*. 238–245.

CHEN, S. AND WILLIAMS, M. A. 2010. Towards a comprehensive requirements architecture for privacy-aware social recommender systems. In *Proceedings of the Asia-Pacific Conference on Conceptual Modelling*. Vol. 110. 33–42.

CHIRITA, P.-A., NEJDL, W., AND ZAMFIR, C. 2005. Preventing shilling attacks in online recommender systems. In *Proceedings of the ACM International Workshop on Web Information and Data Management (WIDM)*. 67–74.

CRAMER, R., DAMGÅRD, I., AND NIELSEN, J. 2001. Multiparty computation from threshold homomorphic encryption. In *Advances in Cryptology – EUROCRYPT*. Lecture Notes in Computer Science, vol. 2045, Springer Verlag, Berlin Heidelberg, 280–289.

DAMGÅRD, I., GEISLER, M., AND KRØIGÅRD, M. 2008a. A correction to efficient and secure comparison for on-line auctions. Cryptology ePrint Archive, Report 2008/321.

DAMGÅRD, I., GEISLER, M., AND KRØIGÅRD, M. 2008b. Homomorphic encryption and secure comparison. *J. Appl. Cryptol. 1,* 1, 22–31.

DAMGÅRD, I. AND JURIK, M. 2001. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In *Proceedings of the International Workshop on Practice and Theory in Public Key Cryptography (PKC)*. 119–136.

DAMGÅRD, I. AND KOPROWSKI, M. 2001. Practical threshold RSA signatures without a trusted dealer. In *Advances in Cryptology – EUROCRYPT*. Lecture Notes in Computer Science, vol. 2045, Springer Verlag, Berlin Heidelberg, 152–165.

DELLAROCAS, C. 2000. Immunizing online reputation reporting systems against unfair ratings and discriminatory behavior. In *Proceedings of the ACM Conference on Electronic Commerce (EC)*. 150–157.

DIERKS, T. AND ALLEN, C. 1999. The TLS Protocol Version 1.0. RFC 2246 (Proposed Standard).

DIERKS, T. AND RESCORLA, E. 2006. The Transport Layer Security (TLS) Protocol Version 1.1. RFC 4346 (Proposed Standard).

DIERKS, T. AND RESCORLA, E. 2008. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard).

DWORK, C. 2006. Differential privacy. In *Proceedings of the International Colloquium on Automata, Languages and Programming (ICALP)*. Lecture Notes in Computer Science, vol. 4052, Springer Verlag, Berlin Heidelberg, 1–12.

DWORK, C. 2008. Differential privacy: A survey of results. In *Proceedings of the International Conference on Theory and Applications of Models of Computation (TAMC)*. Lecture Notes in Computer Science, vol. 4978, Springer Verlag, Berlin Heidelberg, 1–19.

FOUQUE, P.-A., POUPARD, G., AND STERN, J. 2000. Sharing decryption in the context of voting or lotteries. In *Proceedings of the International Conference on Financial Cryptography (FC)*. Lecture Notes in Computer Science, vol. 1962, Springer Verlag, Berlin Heidelberg, 90–104.

GOLDREICH, O. 2004. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press.

GROTH, J. 2010. A verifiable secret shuffle of homomorphic encryptions. *J. Crypt.* 546–579.

GROTH, J. AND ISHAI, Y. 2008. Sub-linear zero-knowledge argument for correctness of a shuffle. In *Advances in Cryptology – EUROCRYPT*. Lecture Notes in Computer Science, vol. 4965, Springer Verlag, Berlin Heidelberg, 379–396.

HOENS, T. R., BLANTON, M., AND CHAWLA, N. 2010a. A private and reliable recommendation system using a social network. In *Proceedings of the IEEE International Conference on Information Privacy, Security, Risk and Trust (PASSAT)*. 816–825.

HOENS, T. R., BLANTON, M., AND CHAWLA, N. 2010b. Reliable medical recommendation systems with patient privacy. In *Proceedings of the ACM International Health Informatics Symposium (IHI)*. 173–182.

KARGUPTA, H., DATTA, S., WANG, Q., AND SIVAKUMAR, K. 2003. On the privacy preserving properties of random data perturbation techniques. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*. 99–106.

KATZENBEISSER, S. AND PETKOVIC, M. 2008. Privacy-preserving recommendation systems for consumer healthcare services. In *Proceedings of the IEEE International Conference on Availability, Reliability and Security (ARES)*. 889–895.

LAM, S. K. AND RIEDL, J. 2004. Shilling recommender systems for fun and profit. In *Proceedings of the ACM International Conference on World Wide Web (WWW)*. 393–402.

LIPMAA, H., ASOKAN, N., AND NIEMI, V. 2002. Secure Vickrey auctions without threshold trust. In *Proceedings of the 6th International Conference on Financial Cryptography (FC)*. 87–101.

MCSHERRY, F. AND MIRONOV, I. 2009. Differentially private recommender systems: Building privacy into the Netflix prize contenders. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (KDD)*. 627–636.

MEHTA, B., HOFMANN, T., AND FANKHAUSER, P. 2007. Lies and propaganda: Detecting spam users in collaborative filtering. In *Proceedings of the International Conference on Intelligent User Interfaces*. 14–21.

MILLER, B., KONSTAN, J., AND RIEDL, J. 2004. Pocketlens: Toward a personal recommender system. *ACM Trans. Inf. Syst. 22,* 3, 437–476.

MOBASHER, B., BURKE, R., WILLIAMS, C., AND BHAUMIK, R. 2006. Analysis and detection of segment-focused attacks against collaborative recommendation. *Adv. Web Mining Web Usage Ana.* 96–118.

PAILLIER, P. 1999. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology – EUROCRYPT*. Lecture Notes in Computer Science, vol. 1592, Springer Verlag, Berlin Heidelberg, 223–238.

PEDERSEN, T. 1991. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology – CRYPTO*. Lecture Notes in Computer Science, vol. 576, Springer Verlag, Berlin Heidelberg, 129–140.

PENG, K. AND BAO, F. 2010. A shuffling scheme with strict and strong security. In *Proceedings of the 4th International Conference on Emerging Security Information, Systems, and Technologies*. 201–206.

PENG, K., BOYD, C., AND DAWSON, E. 2005. Simple and efficient shuffling with provable correctness and ZK privacy. In *Advances in Cryptology – CRYPTO*. Lecture Notes in Computer Science, vol. 3621, Springer Verlag, Berlin Heidelberg, 188–204.

POLAT, H. AND DU, W. 2005. SVD-based collaborative filtering with privacy. In *Proceedings of the ACM Symposium on Applied Computing (SAC)*. 791–795.

SCHOENMAKERS, B. AND TUYLS, P. 2006. Efficient binary conversion for Paillier encrypted values. In *Advances in Cryptology – EUROCRYPT*. Lecture Notes in Computer Science, vol. 4004, Springer Verlag, Berlin Heidelberg, 522–537.

SRIVATSA, M., XIONG, L., AND LIU, L. 2005. Trustguard: Countering vulnerabilities in reputation management for decentralized overlay networks. In *Proceedings of the International Conference on World Wide Web (WWW)*. 422–431.

ZHAN, J., HSIEH, C. L., WANG, I. C., HSU, T. S., LIAU, C. J., AND WANG, D. W. 2010. Privacy-preserving collaborative recommender systems. *Syst. Man, Cybernet. Part C: Appl. Rev. 40,* 4, 472–476.

ZHANG, S., FORD, J., AND MAKEDON, F. 2006. A privacy-preserving collaborative filtering scheme with two-way communication. In *Proceedings of the ACM Conference on Electronic Commerce (EC)*. 316–323.

Received December 2011; revised May 2012; accepted August 2012