

# Dare-to-Share: Collaborative Privacy-Preserving Recommendations with (almost) no Crypto

Pietro Russo, Lorenzo Bracciale, Giuseppe Bianchi

*University of Rome "Tor Vergata" / CNIT, Via del Politecnico, 1 00133 - Rome (Italy)  
{rssptr01, lorenzo.bracciale, giuseppe.bianchi}@uniroma2.it*

---

## Abstract

Collaborative recommending systems aim to predict a potential user-item rating on the basis of remaining ones. Since, in several contexts, sharing of other users' ratings may be prevented by confidentiality concerns, several works have effectively addressed the design of privacy preserving recommenders. Still, most of the proposed solutions rely on advanced cryptographic methodologies, whose may conflict with the simplicity and viability requirements of real world deployments. In contrast, we propose an approach which does not require any complex cryptography. We show that whenever we can tolerate recommendations based on average values, we can transform the recommender into a privacy-preserving one, by using two non colluding replicas of the same system, and by distributing randomly "blinded" data to these replicas. To protecting each user's rating, a key asset of our approach is the ability to conceal which specific items are rated by which users. Our proposal is secure under the honest-but-curious attacker's assumption, and we show how it can be extended to guarantee robustness also against malicious adversaries. Finally, as a proof-of-concept, we present an implementation of the proposed approach for our motivating use case - collaborative assessment of computer/network vulnerabilities without revealing which of them affect one own infrastructure.

*Keywords:* Secret sharing; Verifiable secret sharing; privacy preserving recommending; Pedersen commitment; Risk assessment; Risk management

---

## 1. Introduction

Recommending systems play a pivotal role in the modern internet economy [1], by assisting end users in taking decisions among an essentially very large number of alternatives, and providing Decision Support Systems (DSS) with automated intelligent support. With very few exceptions [2], the majority of recommending systems suggest decisions to users on the basis of a large dataset comprising of ratings (or preferences) given by *other* users.

Most of the recommended techniques addressed in scientific literature leverage quite sophisticated techniques, with regularized matrix factorization, arguably the leading approach since the famous NetFlix context [3] [4]. These techniques are indeed capable of inferring users' preferences/interests from their own rating behavior, and consequently provide diversified recommendations to different user profiles.

### 1.1. What about data privacy?

Recommender system engines are powerful machine-learning software fed by data. As such, the bigger and the more accurate the data is, the more useful its recommendations can be. However, in some contexts, sharing a large data-set of historical data appears to run in stark contrast to users' privacy or business confidentiality needs.

What if the users do not trust the recommender system itself and do not want to share their potentially sensible data with it? Would it be possible to obtain some kind of collaborative service among users without a central trusted authority and generally without anybody, except the user himself, knowing the data?

Even if it appears an apparently ill-posed question, solutions exist and must be sought after in the field of data privacy and security. In that field/community, privacy preserving personalized recommendation services have been studied in literature for a while [5], finding solutions based on sophisticated techniques such as homomorphic encryption [6], garbled circuits [7] and secure multi-party computation [8]. However, despite the growing interest in privacy, the fact remains that practical adoption of privacy preserving systems is still lagging behind.

We propose that the very limited real world deployment of such advanced cryptographic technologies is due to three main reasons.

First, such approaches often require specialized computation such as modular operations on large prime fields which, according to [9], exhibit six orders of magnitude of difference with respect to the time needed to perform regular arithmetic operations in small (32 or 64-bit) fields.

Second, real world system operators are extremely reluctant to trade performance for privacy, as some advanced cryptographic approaches unfortunately would require. As pointed out in [10], practical implementation problems of Secure Multi Party Computation techniques stand in the way of their wider adoption.

Third, most stakeholders may not be capable of digging into the quite technical details of these constructions, and thus may be reluctant to “buy” them without fully understanding their operation. Conversely, understanding technologies is a viable solution to fight the “fear of change” which impairs companies’ innovation [11].

Finally, it is worth noting that it is also very hard to incorporate advanced cryptographic technologies into existing infrastructures and tools, as re-building production-ready implementations is costly and error-prone, and generally not an option for most companies in our fast-paced modern world [9].

## 1.2. Which scenarios?

Many domain-specific applications need privacy-preserving recommendations, think for instance of healthcare, finance, and even day-to-day life [6].

One specific use case which was an original motivation for our work was the opportunity to develop a collaborative risk management system [12], capable of collecting information about *other* companies’ potential and current cyber-security vulnerabilities, and to help an enterprise to evaluate the impact of such threats. However, unless we rely on fully trusted centralized intermediaries, such a system today may hardly exist. Not only are enterprises hardly willing to reveal to other peer companies their evaluation of a given vulnerability, but to a greater extent companies are clearly unwilling to reveal the very existence of *that specific* vulnerability in its domain. We dedicate section 6 specifically to this use case, implementing the privacy preserving mechanism described above, to a collaborative assessment of computer/network vulnerabilities.

### 1.3. Our Contribution

In this paper, our motivating question was: can we design a privacy-preserving collaborative recommending system which uses off-the-shelf arithmetic, performs similarly to a non-protected system, and employs *trivial-to-explain* cryptographic techniques which even a layman may understand? While this appears challenging when considering state of the art recommenders (e.g. based on matrix factorization), in this paper we show that this goal can be met if we consider a basic recommending system whose prediction approach relies on averaging the remaining ratings. Interestingly, despite its simplicity, our proposed system exhibits an excellent level of privacy with a very lightweight cryptographic complexity - our encryption algorithm is merely based on adding a random value to the original data; our implementation relies on (non-prime) small-module arithmetic fully compatible with the word size of commodity CPUs (e.g. 32 or 64 bits), and can be easily understood and mastered by anyone with basic computer science skills.

In terms of scientific contribution, our major challenge was the identification of a solution which, while retaining the computational and deployment simplicity of a trivial secret sharing approach (see section 3.2), could permit us to further conceal which item was rated by whom. This is in fact an important requirement in several scenarios, including the collaborative risk assessment use case presented in section 6. Indeed, hiding the specific vulnerabilities that affect a party (e.g. a company) is arguably even more important than randomizing the score that said party has given to the vulnerability itself.

We accomplish this goal by proposing a novel technical solution based on a binary random matrix, which is used to hide information about the presence or absence of a rating, meanwhile permitting (owing to its further random split between privacy peers) the number of ratings to be counted blindly as this information is necessary for computing the average recommendation. In other words, while a straightforward application of a literature secret-sharing based approach would have only prevented the disclosure of the actual rating, our approach further avoids revealing which item is rated by whom, with negligible supplementary computational complexity. Moreover, we also extend our construction and show how, at the cost of extra computational complexity and with the addition of a tailored cryptographic commitment, the provided solution can also cope with malicious/cheating

parties. In summary:

1. we propose an approach which relies on a very easy and scalable anonymization technique, based on a suitable adaptation of trivial secret sharing techniques which, despite providing unconditional security (inherited from the property of trivial secret sharing), does not require any prime field arithmetic against *honest-but-curious* attackers - more cumbersome cryptography becomes necessary only when the goal is also to protect against *active/malicious* attackers;
2. besides hiding the ratings, our approach introduces a novel technique, namely a "presence matrix" to further hide the presence or absence of the rating itself. This is a crucial property in many contexts, such as the specific vulnerability assessment use case which originally motivated our work, where the information about the existence of a vulnerability is arguably even more important and sensible than its rating;
3. rather than requiring a redesign of existing systems, our approach fosters an incremental deployment and can be applied on top of already implemented systems with minimal adaptation, i.e., by duly randomizing the ratings before transmitting them to the backend servers. In this perspective, current applications can be made "privacy aware" also by adding a proxy and simply duplicating the database on servers belonging to different non-colluding domains, thus not requiring production-ready applications to be rebuilt.

The rest of the paper is organized as follows: section 2 presents the state of the art, section 3 presents the problem definition and the related threat model, section 4 describes the proposed solution, section 5 discusses some limitations of the implemented solution, section 6 presents the case of the implementation of a collaborative cyber-risk assessment service, finally conclusions are drawn.

## 2. Related Work

A recommender system can be defined through four dimensions: real-world application domains, application platforms, recommendation methods and recommender systems software [13]. In the following sections, recommender systems

are analysed not only according to these dimensions, but also considering how they protect data.

### 2.1. Recommendation methods and recommender systems software

Recommender systems are based on two strategies [14]. The first is the *content filtering approach* based on the construction of a profile for every user or product to characterize its nature (this may require gathering external information that might not be available). Another method is the use of past user behaviour (for example, previous transactions or ratings) without requiring the creation of explicit profiles. This approach is *collaborative filtering*. The two primary areas of collaborative filtering are the neighbourhood methods and latent factor models [15]. The Matrix Factorization algorithm is the most successful realization of the latent factor model and has been applied to the Netflix problem [16, 17], a competition to motivate researchers to improve the accuracy of a recommender system. Using this algorithm, it is possible to calculate the most probable choice of a user starting from the other users' choices, without gathering external information on that user.

### 2.2. Privacy preserving recommending systems

Recommender Systems typically require users to reveal their ratings to a recommender service, which subsequently uses them to provide relevant recommendations in terms of predictions [18]. From a privacy point of view, the recommender service act as a trusted third party that collects all the data and then provides recommendations individually to each user. In many cases, however, such an assumption can be criticised. For this reason there are many literature algorithms to create a *privacy preserving* recommending system, i.e. systems that should provide good recommendations but that are not entrusted to understand all the data, at least in a clear form.

The main adopted techniques can be categorized in the following families:

- Based on traditional secure multi-party computation techniques such as homomorphic encryption, Garbled circuit, or special implementation of privacy preserving operations, such as secure sums - representative works in the secure multiparty area include for instance [19, 20, 21]

- Based on a system that trades off privacy with accuracy (e.g. differential privacy work [22] and the referenced papers therein, or through profile obfuscation [23])
- Based on an ElGamal scheme of homomorphic encryption [24] which works as follows: the multiplication of two cypher texts equals the encryption of the multiplication of the plain texts

The key differences of this work with respect to the literature are:

- It is based on just one single communication round among client and server (faster);
- We exploit the diversity of the actors, following an approach similar to SEPIA [25], but all the servers need the same code-base (easy to maintain);
- It is easier to explain and to develop, especially with respect to complex secure multiparty computation techniques such as [21];
- It allows for the easy adoption of existing systems, since it can work without changing code on server side (for each servers) but only with modifications on the client side;
- It implies simply algebraic logic with small non-prime modulus, so it is computationally efficient

### *2.3. Motivations for privacy preserving recommending systems*

The need for privacy preservation is very important for a recommending system, because it needs to protect data shared by the users. Considering the medical word, with the development of the "Internet + Intelligent Medical", where patients can online diagnose some common diseases via the Internet there exist many severe problems on privacy for medical sensitive data of patients. Many solutions have been proposed to solve this problem such as privacy-preserving self-serviced medical diagnosis scheme based on secure multi-party computation [26].

In a financial context, the wide diffusion of cloud computing has caused an increase in the number of restrictions that financial firms apply to cloud applications [27]. To this aim, many solutions have been proposed and implemented to protect financially sensitive data against unauthorized access[28, 29, 30]. One of the most recent one is the developing of a cryptographic-based model sharing

technique to securely outsource knowledge reflected in decision trees of multiple parties, designing a secure computation mechanism to facilitate privacy-preserving knowledge transfer [31].

The need for privacy preservation is very important also in the field of social networks, where billions of users share their personal data with various devices over the Internet on a daily basis [32, 33]. This need is a primary target not only for the traditional and famous social networks (Facebook, Instagram, etc..) but also for different types of social networks such as the Diaspora network [34], which was a decentralized online social network with over 216000 users. It is a network of independent, federated Diaspora servers that are administered by individual users who allow Diaspora users' profiles to be hosted on their servers.

The particular use case presented in this paper comes from the world of Cyber Risk Management, where there is an important need of privacy or business confidentiality.

#### *2.4. Application domains and application platforms*

Recommending systems use assistant mechanisms for decision-making. For this reason, there are many fields where these types of systems are used, such as movie or music recommendation [35, 36] or the prediction of Chronic Hepatitis B (CHB) in patient clinical medication [37]. In the last years with a greater use of devices connected to internet, recommending systems have increased their importance also in sectors such as e-government, e-business, e-commerce/e-shopping, e-library, e-learning, e-tourism, e-resource services and e-group activities [13] with the aim of providing users with personalized online recommendations to handle the increasing information overload problem and improve customer relationship management.

Moreover, a recommender system can provide a viable solution to the problem of the exponential increase of available information (information overload). This is done by helping users to find relevant information, for instance by using fuzzy linguistic recommender systems to advise researching resources in university digital libraries [38, 39]. This work, and in particular the use case presented in section 6, can provide a practical solution to such a problem by suggesting users reasonable values for an application-specific domain, and doing this in a privacy preserving



form.

### 2.5. Literature summary

Considering the related works cited before, a summary with the most relevant ones is presented in the comprehensive table 1:

Reference	Cryptographic method	Recommender method
[3] [4]	No cryptography	Matrix Factorization and neighbor based algorithms
[6]	Homomorphic encryption (based on ElGamal scheme)	Content-based filtering and collaborative filtering recommendations
[7]	Garbled circuits	Matrix Factorization
[8]	Secure multi-party computation	Decision trees
[15]	No cryptography	Collaborative Filtering (based on Genetic algorithms)
[20]	Data perturbation techniques	Item-based collaborative filtering algorithm
[24]	Homomorphic encryption (based on ElGamal scheme)	Scalar Product

Table 1: Comparative analysis of the cryptographic and recommending methods presented in the related work

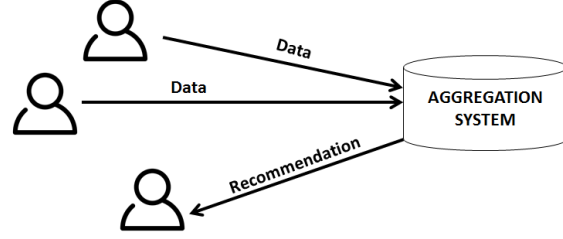
## 3. Problem statement and baseline approach

We consider a set of  $n$  users that rate a subset of  $m$  possible items (e.g. movies, risks). Users want to receive a recommendation for the items they did not vote for whilst keeping their own votes secret.

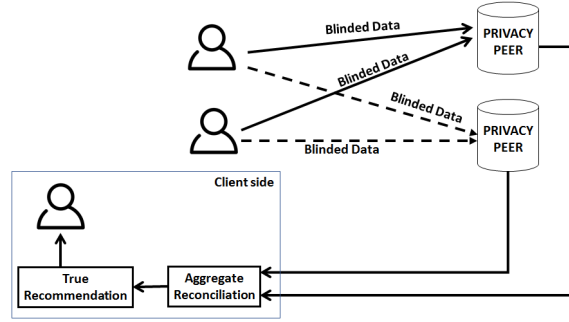
The conventional approach to address this problem is to use a Recommended System (RecSys). As represented in figure 1.a, the RecSys usually is a centralized entity and represents, from a security point of view, a Trusted Third Party (TTP) for the whole system.

### 3.1. From TTP to Privacy Peers

Entirely relying on a Trusted Third Party is a strong assumption and represents an important security bottleneck that should be avoided whenever possible, especially if we are dealing with multiple companies and very sensitive data. For this



(a)



(b)

Figure 1: (a) presents the conventional architecture of a Recommending System, whereas (b) is an example of a 2 peers distributed architecture

reason, starting from the approach pursued by works such Security through Private Information Aggregation (Sepia) [25] and Peer for Peer (P4P) [40], alternative solutions have been presented in literature to relax the trust assumption and replace the need for a single Trusted Third Party with a “Simulated TTP” composed of a set of non-colluding semi-honest servers that we call *Privacy Peers*. In this way, we move the trust assumption from the full honesty of a single entity (TTP) to the non-collusion assumption of a set of different entities (Privacy Peers) which is a more reasonable requirement in systems operated by multiple independent entities. The general architecture proposed is depicted in figure 1.b: users generate “blinded data” from their original data and commit it to the privacy peers that recommend over this randomized data. The peers then publish the aggregated results and a public data reconciliation allows the peers to obtain the true recommendations.

### 3.2. How to “blind data”: Trivial Secret Sharing

Suppose we want to store a secret number  $s$  on two privacy peers, namely peer A and peer B, such that neither is able to independently infer any information on this number, but  $s$  can be reconstructed only when both entities are involved. Unlike Shamir’s classical (and more general) secret sharing problem [41], the above described scenario does not require any specific cryptographic skills, and can be addressed using a trivial approach - indeed the approach described in what follows is often named “*trivial secret sharing*”.

The idea is to generate a random number  $r \in [0, N - 1]$  and then send  $r$  to peer A and  $x = s - r \bmod N$  to peer B. Note that  $N$  can be a small non-prime number: the only requirement is that  $N$  should be as large as the maximum possible value of the secret  $s$ . Clearly, peer A, which received the random value  $r$ , has no information about  $s$ . But the same holds for peer B, as the modular operation  $x = s - r \bmod N$  is conceptually equivalent to an unconditionally secure one-time pad encryption of the value  $s$  - in other words, it is straightforward to show that an adversary who observes  $x$  has no advantage in guessing the secret  $s$ : since  $r$  is a random quantity uniformly distributed in the range  $[0, N - 1]$ , all possible guesses for  $s$  in the range  $[0, N - 1]$  become now equally likely. However, the secret  $s$  can be trivially reconstructed as long as peers A and B share their values. Indeed, to reconstruct  $s$ , it suffices to sum (modulus  $N$ ) the “share”  $r$  provided by peer A with the “share”  $x = s - r \bmod N$  provided by peer B, i.e.,  $s = r + (s - r) \bmod N$ . What is appealing about such an elementary technique is that operations are performed using small non-prime modulus, see for instance the example shown in figure 2 ( $s=2$ ,  $N=5$ ,  $r=3$ ) which uses modulus  $N = 5$ .

A compelling property of trivial secret sharing (as well as many other more general secret sharing techniques, including Shamir’s and mainstream secret-sharing-based secure multiparty frameworks such as Sepia [25] and P4P [40]) is linearity with respect to the sum operation. If we need to sum an arbitrary number of secrets (e.g. ratings), a share of such a total can also be computed by summing the shares of each individual rating. This “homomorphic” property can be used to build a private recommending system where the recommendation is based on a simple average value of the scores.

In particular, if we consider  $n$  users, any one of which votes a set of  $m$  items,

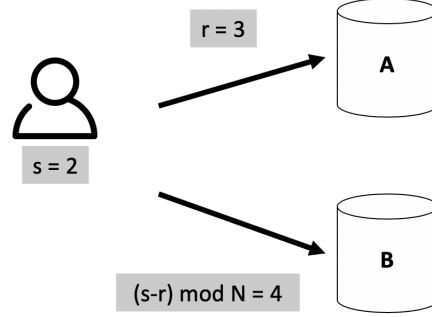


Figure 2: Trivial Secret Sharing with  $s=2$ ,  $N=5$ ,  $r=3$

we just need to move from the single secret value  $s$  to a vector of secret values  $\mathcal{S}_i$  which contains all the  $m$  scores given by the  $i$ -th user. We call  $s_{ij}$  the score given by the  $i$ -th user to the  $j$ -th item. In the same way, we have to pass from a single random variable  $r$  to a vector of random variables  $\mathcal{R}_i$  containing  $m$  random values that the  $i$ -th user will send to peer A. The random value sent by the  $i$ -th user is called  $r_{ij}$ , and it is associated with the  $j$ -th item. After users send their vectors  $\mathcal{R}_i$  and  $\mathcal{S}_i - \mathcal{R}_i$ , each privacy peer builds a matrix by combining the received vectors: peer A has  $\mathcal{R}$ , a  $m \times n$  matrix with all the random values sent by all the users and regarding all the items, while peer B has  $\mathcal{S}_i - \mathcal{R}_i$ , a  $m \times n$  matrix with the differences modulo  $N$ . Now, we can compute the sum of the ratings for each item by exploiting the homomorphic property as follows. Peer A computes the column-wise sum of its matrix  $\mathcal{R}$ , while peer B computes the column-wise sum of its values  $\mathcal{S} - \mathcal{R}$ . If both the peers advertise these sums, every user can easily obtain the overall sum of all the secrets by simply computing  $\text{sum}(\mathcal{S}) = \text{sum}(\mathcal{R}) + \text{sum}(\mathcal{S} - \mathcal{R})$ .

### 3.3. Challenge: from sums to average ratings

The problem now is the following: how it is possible to pass from a sum of ratings to the average of them, which is the aim of our simple recommender. If we knew the number of rates per item, this would be trivially obtained by dividing the sum by this number. However, the number of raters is *not*  $n$  (not all the users vote for all the items). Moreover, we also want to avoid keeping track of whether a single user voted or did not vote for a given item. Providing such information requires

special attention, since it can be very sensitive in many real-world application scenarios such as in the case of security assessment (described in section 6) where we clearly do not want to share publicly the impact values of the single vulnerabilities on our system, nor do we want to share which vulnerability is applicable to our company.

This problem cannot be addressed by using a more general secret sharing framework (e.g. Sepia [25] or P4P [40]), but requires a tailored solution. To this aim we introduce a *presence matrix*, namely  $\mathcal{P}$ , that is a binary  $n \times m$  matrix revealing whether a user voted for an item or not. As for the case of the scores, we protect this matrix too, extending the approach from  $\mathcal{S}$  to  $\mathcal{P}$ . The complete solution is described in the following section.

### 3.4. Threat model

Users want to keep their data secret from: i) other users; ii) malicious recommender systems; iii) third party attackers able to violate recommender system infrastructure.

The above threats can be compressed in two attack models:

- **Honest-but-curious parties:** all parties follow the protocol honestly.
- **Malicious parties (dynamic adversary):** not only malicious users (can commit bogus data for the sake of revealing other users' data) but also malicious RecSys.

We do not assume the presence of a trusted third party. Conversely, we adopt a distributed strategy assigning the recommending task to multiple independent parties, called *privacy peers*, that can be physically hosted on different companies. We require non-colluding privacy peers.

### 3.5. Security guarantees

The system, in its basic form, offers protection against honest-but-curious parties: all parties learn nothing about users' scores both in terms of presence and in terms of actual value. With the addition of commitments (see section 4.3), we offer the possibility for users to commit their data to detect data changes by peer

nodes and thus to cope with malicious peers. The system will also continue offering information on theoretical security in the case of malicious users, but we did not design the system to resist attacks aimed at corrupting data. Dedicated access control or specific mechanisms to prevent Sybil attacks can be used for this purpose [42].

## 4. Proposed solution

### 4.1. Dare-to-Share approach

We consider a set of  $n$  users and  $m$  items.

We describe the approach with reference to the  $i$ -th user holding a set of secret scores for a subset of the  $m$  items described by a sparse vector  $S_i \in \mathbf{Z}^m$ , and with reference to two privacy peers, namely peer A and B. Each score is in the range 0 to  $MAX\_SCORE\_ITEMS$ .

The goal is to provide the user with suggestions about missing values according to the average score given by the other users.

To this aim, the Dare-to-Share approach results in the following operations:

- **Step 1** User 0 builds a secret vector  $S_i \in \{0, MAX\_SCORE\_ITEMS\}^m$  and starting from that, the presence binary vector  $P_i \in \{0, 1\}^m$  which specify whether the user voted for a given item.
- **Step 2** User builds two vectors  $\mathcal{R}_i \in \{0, MAX\_SCORE\_ITEMS\}^m$  and  $Q_i \in \{0, 1\}^m$ . Both vectors contain pure random numbers. Then the user sends  $\mathcal{R}_i$  and  $Q_i$  to peer A, and  $(S_i - \mathcal{R}_i) \bmod \mathcal{M}_r$  and  $(P_i - Q_i) \bmod 2$  to peer B.
- **Step 3** All the peers compute the column-wise sum of the received matrix ( $\mathcal{R}$  and  $Q$  for peer A, and  $S - \mathcal{R}$  and  $\mathcal{P} - Q$  for peer B) and publicly advertise the resulting vectors.
- **Step 4** Taking the vectors from the previous step, everybody can compute the average values for all the scoring items as follows. Calling  $V_X$  the vector derived from the generic matrix  $X$ , we see that:

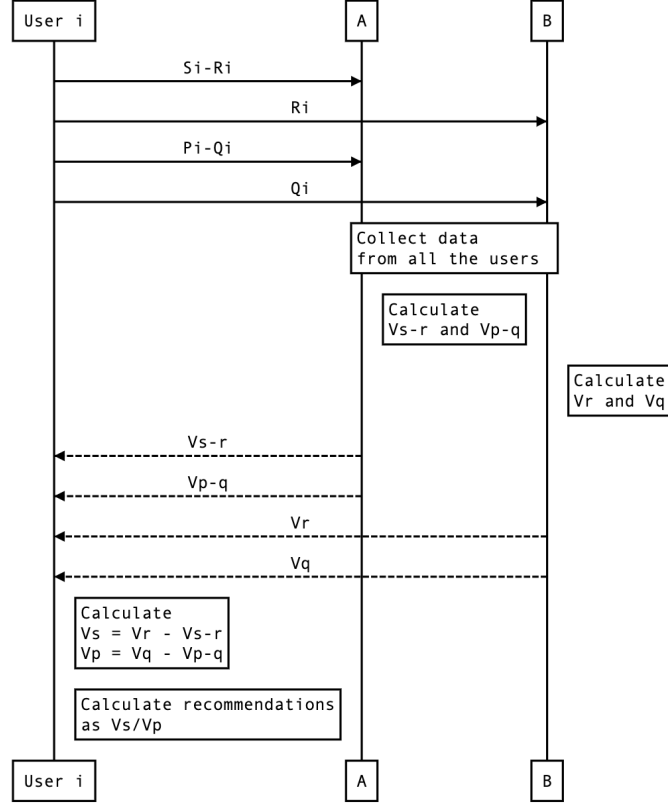


Figure 3: Sequence diagram of the Dare-to-Share approach. Dashed lines represent advertised data (from a privacy peer to everybody), solid lines represent private data transfers (from user to a privacy peer). All calculations use modular arithmetic.

- 1)  $V_R + V_{S-R} \bmod \mathcal{M}_s$ <sup>1</sup> representing a vector containing the sum of all the scores that users give to a particular item.
- 2)  $V_Q + V_{P-Q} \bmod \mathcal{M}_p$ <sup>2</sup> containing the number of users that have voted for a particular item.
- 3) Their ratio is equal to the average value scores for all the items.

Figure 3 summarizes in a sequence diagram all the Dare-to-Share operations.

<sup>1</sup> $\mathcal{M}_s$  is an integer greater than  $m \times \text{MAX\_SCORE\_ITEMS} + 1$

<sup>2</sup> $\mathcal{M}_p$  is an integer greater than  $m \times \text{MAX\_SCORE\_PRESENCE} + 1$

#### 4.2. Numerical example

We provide a numerical example considering a system with 2 users, 2 peers and 2 items with the secret matrix  $\mathcal{S}$  shown in table 2:

	Item1	Item2
User1	4	-
User2	2	3

Table 2: Example of matrix  $\mathcal{S}$

The associated presence matrix  $\mathcal{P}$  is shown in table 3:

	Item1	Item2
User1	1	0
User2	1	1

Table 3: Example of matrix  $\mathcal{P}$

We define the following values:

- $m$  is the number of users (2 in this case);
- MAX\_SCORE\_ITEMS is the maximum value for the elements in  $\mathcal{S}$  (in this example, 5);
- MAX\_SCORE\_PRESENCE is the maximum value for the elements in  $\mathcal{P}$  (and equal to 1 by definition);
- $\mathcal{M}_s = 11$ ;
- $\mathcal{M}_p = 3$ ;

Then we generate the random matrices  $\mathcal{R}$  (in table 4) and  $\mathcal{Q}$  (in table 5) :

	Item1	Item2
User1	3	1
User2	4	0

Table 4: Example of matrix  $\mathcal{R}$



	Item1	Item2
User1	2	1
User2	0	1

Table 5: Example of matrix  $\mathcal{Q}$

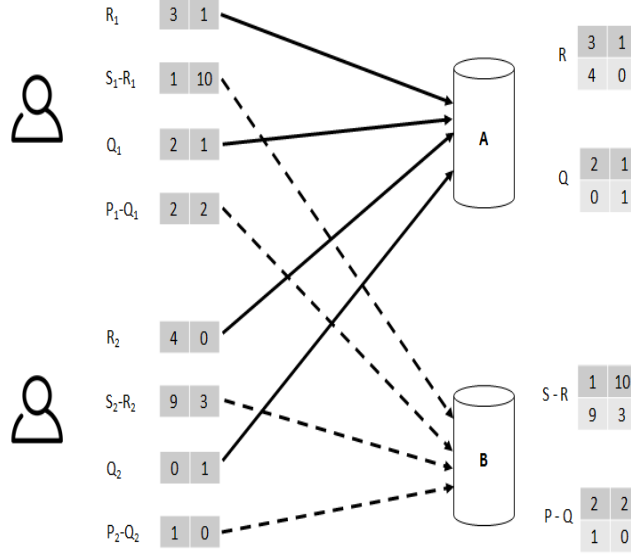


Figure 4: Example message exchange in the case of 3 users and 2 peers

Using the notation of  $\mathcal{X}_y$  to indicate the  $y$ -th row of matrix  $\mathcal{X}$ , each user proceeds to send their vectors to the privacy peers A and B as shown in figure 4.

At the end of this process, peer A will have tables  $\mathcal{R}$  and  $\mathcal{Q}$ , peer B will have tables  $\mathcal{S} - \mathcal{R}$  and  $\mathcal{P} - \mathcal{Q}$ . Importantly, neither of them have information about the original secret  $\mathcal{S}$ . By combining the columns of these matrices, can we calculate the average values for each rated item. For example, the average values of the first item  $\mathcal{A}_1$  is given by:

$$\mathcal{A}_1 = \frac{\left[ \sum_{j=1}^m R_{j1} + \sum_{j=1}^m (S - R)_{j1} \right] \text{mod} M_s}{\left[ \sum_{j=1}^m Q_{j1} + \sum_{j=1}^m (P - Q)_{j1} \right] \text{mod} M_p} \quad (1)$$

Which corresponds in this example to:

$$A_1 = \frac{[(4+3) + (9+1)] \bmod 11}{[(2+0) + (2+1)] \bmod 3} = \frac{6}{2} = 3 \quad (2)$$

#### 4.3. Extension against malicious party

So far we have described the basic approach that has the great advantage of being simple and, from a system requirement point of view, extremely efficient since it uses ordinary arithmetic without adding extra complexity other than requiring the involvement of multiple parties.

We will show in section 6.3 how that approach is extremely lightweight in terms of resource consumption while guaranteeing excellent privacy coverage against the honest-but-curious attacker model. However, if extra protection is needed to cope with potential malicious parties, we can use modular prime cryptography to make secret sharing *verifiable*, for instance leveraging conventional literature techniques such the Pedersen commitment [43].

We remark that our particular application cannot afford any non-perfectly hiding commitment (such as Feldman's commitment scheme [44]). Since scores are typically small integers, an enumeration attack would be disastrous.

In the following, we describe the commitment procedure with reference to the secret matrix  $\mathcal{S}$ . The same procedure must also be repeated for the presence matrix  $\mathcal{P}$ .

**User side:** Each user commits *every single element* of the matrix  $\mathcal{S}$ , for instance by  $s_{ij}$ , by computing  $h^{t_{ij}}g^{s_{ij}}$  where  $g, h$  are (public) generators,  $t_{ij}$  is a (large) random number, and all operations are done modulo  $p$  (large prime). The result is publicly advertised. Then each user generates a random value  $k_{ij}$  and sends  $k_{ij}$  to peer A and  $t_{ij} - k_{ij}$  to peer B.

**Peer side:** Each peer advertises the sum of the received data aggregating by item. Thus peer A advertises  $V_K$  whose  $i$ -th element is obtained by  $\sum_{j=1}^m k_{ij}$ , while peer B advertises  $V_{T-K}$  whose  $i$ -th element is obtained by  $\sum_{j=1}^m t_{ij} - k_{ij}$ .

**Verification:** The verification is based on the availability of the users' commitments and of the values of  $V_K, V_{T-K}$  together with  $V_R$  and  $V_{R-S}$  (the column-wise sum vectors described in step 4 of section 4.1). The commitment verification procedure is the following: i) multiply all users' commitments for a given item; ii) compute  $g^b h^a$  where  $a$  is the  $j$ -th element of vector  $V_K + V_{T-K}$  and  $b$  is the  $j$ -th

element of vector of  $V_R + V_{S-R}$ ; iii) compare these two results. The commitment is verified if the two values are the same.

This commitment is perfectly hidden, and computational binding can detect any cheating by the parties, although it does not prevent users from committing fake (out of range) data.

We remark that this solution, from a performance point of view, is totally reliant on the clients, thus representing an extra (optional) protection that can be provided on top of the basic approach.

## 5. Discussion

As discussed in previous sections, in this paper we present a mechanism to guarantee data privacy that can be easily and effectively implemented in privacy-preserving support systems. Our main goal is not to present a new recommender system or a new decision support system, but we want to show how, by changing the architecture and adopting easy-to-explain basic techniques, we can empower legacy (and new) systems with data privacy, i.e. nobody (not even servers) knows the data collected from the users of the system. Obviously, the consequences of this simplicity are the two big limitations of this mechanism: the recommending algorithm and the possibility of collecting data to test the algorithm.

The first limitation is that the recommending algorithm used is the average. Notwithstanding its simplicity, it is fair to remark that many real world deployments rely on basic recommendation techniques, such as the one that we will specifically discuss throughout this paper. For instance, customers of advisory sites even for quite subjective choices (e.g. restaurants or hotels, including major players such as TripAdvisor, Booking.com or even Google) are often provided at least at the start of their selection process with a summary rating computed as the average of *all* the received ratings per item. This average value is often further represented in a more intuitive “star system”.

The second limitation is the availability of data to test the algorithm and its performance strongly. For example, considering the use case presented in section 6, only about 25% of records saved in DBs come from real network scans, with the remaining randomly calculated from the scanned data. In the future we plan

to increase the amount of real data by using scans from different organizations' networks, and enlarging the input fields in order not only to cover vulnerability scans, but also other categories that can be found in a fully-fledged risk assessment tool.

## 6. Use case: Risk Assessment

In this section we describe CYRVM (Cyber Risk Vulnerability Management), a custom-made software platform devised to simplify and improve automation and continuity in cyber-security assessment. This software helps network administrators to perform *collaborative* risk assessment in a privacy preserving manner, using the methodology previously described.

### 6.1. Problem description

In recent years, with an ever increasing use of online services and a greater distribution of connected devices, the evaluation of cyber-security risks has gained importance [45, 46]. This has been fuelled also by the many recent data breaches hitting the headlines, such as Marriot, Quora, Cambridge Analytica, to name a few.

Risk assessment is also necessary for any organization to comply with the EU General Data Protection Regulation (GDPR) which states that each data controller must conduct risk checks on a regular basis.

Generally, we can define risk as a measurement of the extent to which something is threatened by an event. Risk is thus related to:

- The consequence of the event on that resource of the company (*impact*);
- The probability of the event (*likelihood*).

Traditionally, this problem has been analyzed by companies' security officers following the risk management procedure of the National Institute of Standards and Technology (NIST) [47]. This implies the identification, evaluation and prioritization of risks followed by the application of instruments to minimize, monitor and control the probability of unfortunate events.

This operation involves a great responsibility often demanded of single experts, who might easily fail in under- or overestimating the importance of an asset, how dangerous a data loss could be, or what type of cyber attack could affect them.

For this reason, the last few years have seen an increase in the number of tools, predictive modelling and machine learning techniques applied to risk analysis [48, 49] and cyber risk management [50, 51, 52] with the aim of predicting risk and contrasting cyber crimes [53]. An idea to improve the quality of prediction is to use a collaborative approach [54, 55, 56, 57, 58] with a *Trusted Server* where users store their security assessment information in order for it to be used to calculate the prediction.

Our idea goes in this direction, offering a solution for assessing the right risk value for every asset and vulnerability by suggesting and predicting reasonable values obtained from a collaborative but privacy preserving system. The main difference of our approach is the capability of predicting user choices without the need of a *Trusted Server* and of protecting users from revealing not only the actual score values submitted to the system, but also which items are evaluated, which would reveal information about the presence of a vulnerability on the system.

## 6.2. Implementation

We implemented the platform as a web application, which works as follows.

Users access to the service as a conventional web page, i.e. by downloading a static web page and the related assets. A javascript code, acting on the client side, performs the encryption logic and handle the communication with the privacy peers. Figure 6 depicts this scenario with the case of two privacy peers, namely A and B.

Privacy peers A and B host a database with two tables containing the *Impact* and *Likelihood* information. In addition to these tables, we introduced two more tables, namely *presence Impact* and *presence Likelihood*, which keep track of the presence of a certain vulnerability on the host. Finally, there is also a public common table containing the list of all vulnerabilities. An excerpt of this table is shown in figure 5.

The platform works as follows:

1. The user uploads a vulnerability scan. This file can be obtained from vulnerability scanning software such as OpenVAS [59] or OWASP Zap.
2. The system proposes, for each vulnerability, a rate in terms of impact and likelihood, which is obtained from the average value given by the other users'

V2	TCP timestamps
V3	OS End Of Life Detection
V4	Microsoft Windows SMB Server Multiple Vulnerabilities
V5	DCE/RPC and MSRPC Services Enumeration Reporting
V6	Acme thttpd and mini_httpd Terminal Escape Sequence in Logs Command Injection Vulnerability
V7	SSL/TLS: Report Vulnerable Cipher Suites for HTTPS
V8	SSL/TLS: Certificate Signed Using A Weak Signature Algorithm
V9	ICMP Timestamp Detection
V10	OS Detection Consolidation and Reporting
V1	Acme thttpd and mini_httpd Terminal Escape Sequence in Logs Command Injection Vulnerability
V11	GSA Default Admin Credentials
V12	SSL/TLS: Diffie-Hellman Key Exchange Insufficient DH Group Strength Vulnerability

Figure 5: Example of vulnerabilities table

choices for the same vulnerability.

3. The user may:

- **Accept the prediction.** In this case the user agrees with every predicted value and does not want to apply any change. The system sets 1 into the *presence* table.
- **Follow the others.** In this case the user wants to follow the recommendation, e.g. because of a lack of knowledge about that specific vulnerability impact assessment. This case is different from the previous ones, since the user passively accepts the calculated values but does not want to vote for that item. The system sets 0 into the *presence* table.
- **Modify your values.** In this case the user decides to change some values (typically affecting the element where they are more confident), and leaving the other values to the default prediction given by the software. The system sets 1 and 0 into the *presence* table accordingly.

The program interface is shown in figure 7 and an example of a result in figure

8.

In figure 9 a) we present the workflow of the algorithm implemented into the

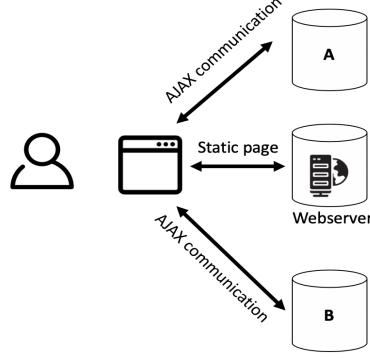


Figure 6: Risk Assessment Architecture: the secret sharing computation is performed on the client side in the browser













CYRVM system. The first step is the upload of the OpenVAS scan result. This scan contains all the vulnerabilities of the network and their severity and likelihood. This data is “blinded” (as described in section 3) and sent to Privacy Peers A and B together with the related presence matrices. Servers update their tables adding an extra row to accommodate the new system, and expanding the previous tables to fit the potential new vulnerabilities received.

Then, every privacy peer calculates the sum (in module) of every column of the Impact, Likelihood, Presence Impact and Presence Likelihood tables, and communicates these values to the user. At this point the user has 8 values for every vulnerability. The user then performs the pairwise modular sums (ImpactA + ImpactB, LikelihoodA + LikelihoodB, etc..) and calculates the difference between Impact and Presence Impact and between Likelihood and Presence Likelihood. Once here, the values of Impact and Likelihood are presented to the user, who can accept or update them. The new values of Impact, Likelihood and Presence (1 or 0) are secret. Starting from these values, the user generates the vectors  $\mathcal{R}_i$  and  $\mathcal{S} - \mathcal{R}_i$  and saves them into the servers A and B.

In figure 9 b) we present the case of a user who decides to read or update their values. In this case, a request is sent to the privacy peers, which calculate, as shown in figure 9 a), the values of the modular sum of every column of the Impact, Likelihood, Presence Impact and Presence Likelihood tables. In addition to these 8 values, rows of the requested system are sent. After that, the user has

Assets table

SHOW 10 ENTRIES SEARCH:

Name	Device Label	Type	Description	Docs	Users	System Mission	Integr.	Avail.	Confid.	Value	Controls
192.168.1.1	192.168.1.1		Inserted from a OpenVAS report			To be defined	3	3	3	05	 
192.168.1.18	192.168.1.18		Inserted from a OpenVAS report			To be defined	3	3	3	05	 
192.168.1.2	192.168.1.2		Inserted from a OpenVAS report			To be defined	3	3	3	05	 
192.168.1.3	192.168.1.3		Inserted from a OpenVAS report			To be defined	3	3	3	05	 
192.168.1.5	192.168.1.5		Inserted from a OpenVAS report			To be defined	3	3	3	05	 
192.168.1.7	192.168.1.7		Inserted from a OpenVAS report			To be defined	3	3	3	05	 
Name	Device Label	Type	Description	Docs	Users	System Mission	Integr.	Avail.	Confid.	Value	Controls

(a)

NET9						
VULN NAME	TCP timestamps	OS End Of Life Detection	Microsoft Windows SMB Server Multiple Vulnerabilities	DCE/RPC and MSRPC Services Enumeration Reporting	SSL/TLS: Report Vulnerable Cipher Suites for HTTPS	GSA Default Admin Credentials
IMPACT	3	2	5	4	3	3
LIKELIHOOD	4	4	4	3	4	3

+ UPDATE  
IF YOU ARE AN EXPERT AND YOU ARE COMPLETELY AGREE WITH THE ABOVE CHOICES\*

+ UPDATE YOUR CHOICES  
IF YOU JUST WANT TO UPDATE YOUR PREFERENCES\*

+ FOLLOW THE OTHERS  
IF YOU JUST WANT TO FOLLOW THE OTHER'S CHOICES\*

(b)

Figure 7: (a) presents all assets of a selected network. (b) presents the values of Impact and Likelihood for every vulnerability of the network.

all the information needed to rebuild the secrets. Values of Impact and Likelihood are presented to the user only if the corresponding bit of the presence tables are 1, implementing a mask on the user interface.

### 6.3. Performance assessment

To analyse the performance of the system, we compared CYRVM with an equivalent system without the privacy preserving mechanism, with the goal of assessing the overhead in terms of resource required on the client side.

In the rest of the chapter we will refer to the privacy preserving algorithm as **Privacy Algorithm** and the other as **Clear Algorithm**.

We ran our tests on a PC with the following features:



Vulnerability Name:	Microsoft Windows SMB Server Multiple Vulnerabilities-Remote (4013389)
Description:	CVE-2017-0143,CVE-2017-0144,CVE-2017-0145,CVE-2017-0146,CVE-2017-0147,CVE-2017-0148 cvss_base_vector=AV:N/AC:M/Au:N/C:C/I:C/A:C summary=This host is missing a critical security update according to Microsoft Bulletin MS17-010. vulndetect=Send the crafted SMB transaction request with fid = 0 and check the response to confirm the vulnerability. insight=Multiple flaws exist due to the way that the Microsoft Server Message Block 1.0 (SMBv1) server handles certain requests. impact=Successful exploitation will allow remote attackers to gain the ability to execute code on the target server, also could lead to information disclosure from the server. Impact Level: System affected= Microsoft Windows 10 x32/x64 Edition Microsoft Windows Server 2012 Edition Microsoft Windows Server 2016 Microsoft Windows 8.1 x32/x64 Edition Microsoft Windows Server 2012 R2 Edition Microsoft Windows 7 x32/x64 Edition Service Pack 1 Microsoft Windows Vista x32/x64 Edition Service Pack 2 Microsoft Windows Server 2008 R2 x64 Edition Service Pack 1 Microsoft Windows Server 2008 x32/x64 Edition Service Pack 2 solution=Run Windows Update and update the listed hotfixes or download and update mentioned hotfixes in the advisory from the below link, <a href="https://technet.microsoft.com/library/security/MS17-010">https://technet.microsoft.com/library/security/MS17-010</a>  solution_type=VendorFix qod_type=remote_active
Threat Source:	Outsider
Threat Action:	Exploit vulnerabilities on internal organizational information systems.
Assets Involved:	192.168.1.5
Notes:	
Current Controls:	No Current Controls defined.
Planned Controls:	No Planned Controls defined.
Likelihood Score:	0,6 out of 1.0
Impact Score:	9,3 out of 10
Risk Score:	56 out of 100
Recommended Controls:	No Recommended Controls defined.

Figure 8: CYRVM result for a particular vulnerability

- Intel Core i7-3632QM;
- CPU 2.20GHz 8;
- 8GB DDR3 Memory;
- Operating System: Ubuntu 16.04 LTS 64bit;
- Apache Server version 2.4.18;

We analyzed the performance of the algorithms in two different types of operations: the case of a new network (i.e. new data inside the system) and the case of updating existing data. This is due to the number of operations to execute - far more in the case of a new network. Figure 10.a presents percentages of use of CPU and memory when inserting a new network. As we can see, the *Privacy Algorithm* needs a higher yet contained amount of resources than the *Clear Algorithm*, with

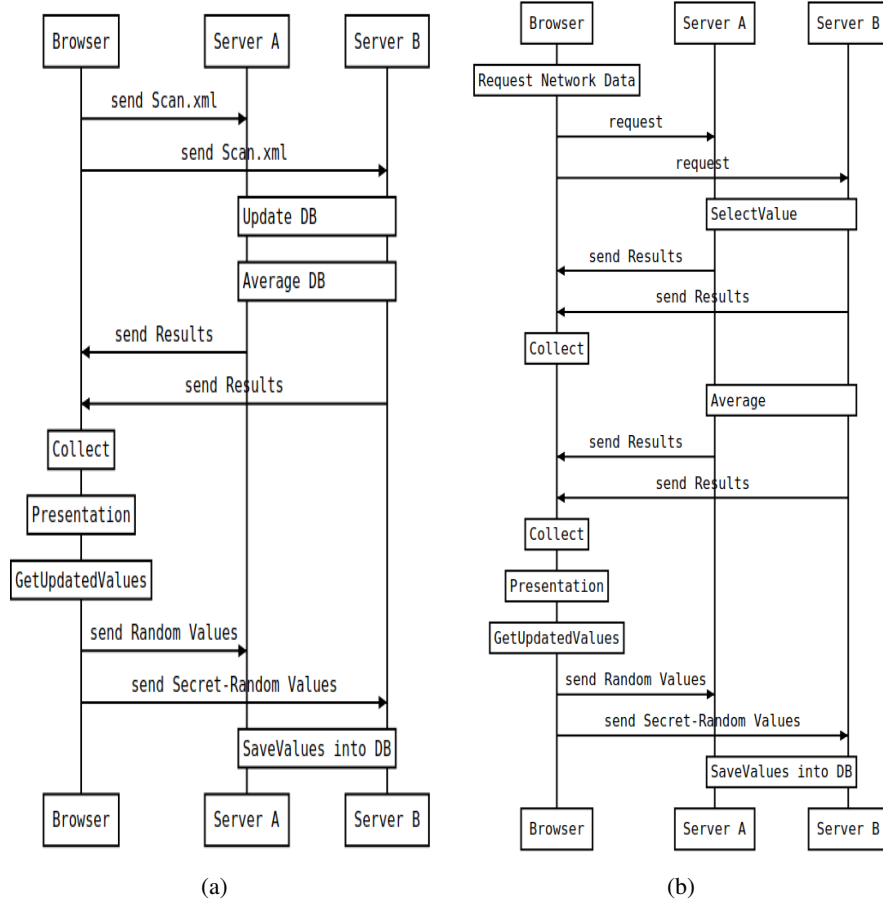


Figure 9: In (a) Secret Sharing work flow. In (b) Secret Sharing work flow when a value is updated

a medium response time of 20 seconds compared to the 12 seconds of the *Clear Algorithm*.

In figure 10.b we present percentages of use of CPU and memory in case of updating an existing network. We used the same DB defined above, where for each network we changed only two values. Also, in this case the *Privacy Algorithm* needs more resources than the *Clear Algorithm*. What emerges from this analysis is that the proposed privacy preserving algorithm, besides the presence of multiple peers, only needs slightly more resources with respect to the clear algorithm on the client side, making it a suitable solution also from a system requirements point of

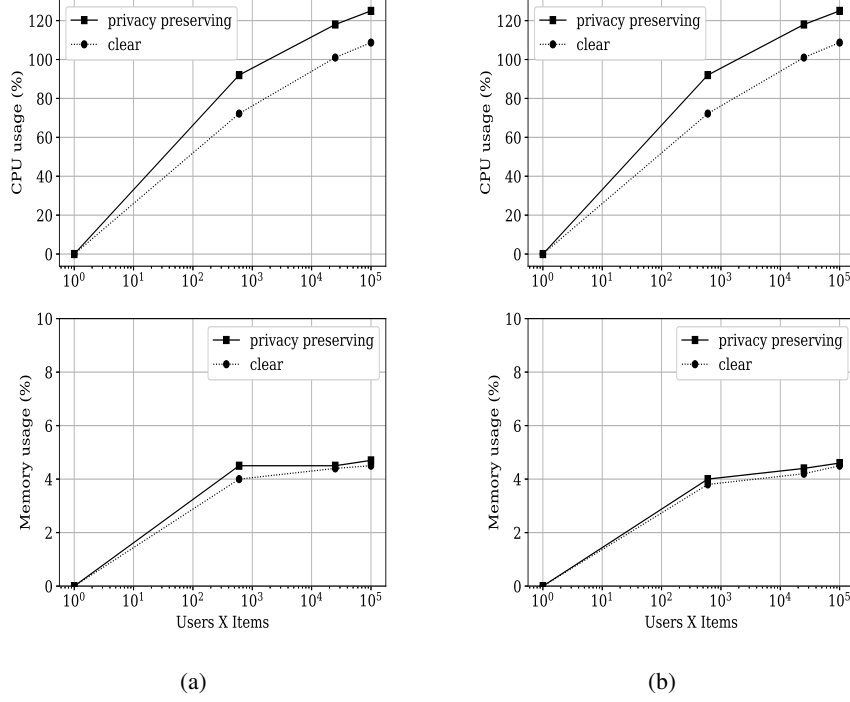


Figure 10: In (a) usage percentages when adding a new network, and (b) when updating an existing network

view.

## 7. Conclusion

In this work we presented a simple architecture that fosters privacy preserving collaboration. We designed a system able to provide the average values of ratings given by a different user to different items, without requiring any Trusted Third Party. The system has the advantage of being very easy to explain and to implement, other than being unconditionally secure, hiding the presence of both the submitted elements and their values. We presented a practical use case together with a working implementation, showing how the solution can be applied in the field of risk assessment.

**Acknowledgements**

This work has been partially funded by the BPR4GDPR project from the European Union's Horizon 2020 research and innovation programme under grant agreement No 787149.

**Data Availability**

The data that support the findings of this study are available on request.

## References

- [1] I. MacKenzie, C. Meyer, and S. Noble, “How retailers can keep up with consumers,” *McKinsey & Company*, 2013.
- [2] M. Fleischman and E. Hovy, “Recommendations without user preferences: a natural language processing approach,” in *IUI*, vol. 3. Citeseer, 2003, pp. 242–244.
- [3] J. Bennett, S. Lanning *et al.*, “The netflix prize,” in *Proceedings of KDD cup and workshop*, vol. 2007. New York, NY, USA., 2007, p. 35.
- [4] G. Takacs, I. Pilaszy, B. Nemeth, and D. Tikk, “Matrix factorization and neighbor based algorithms for the netflix prize problem,” in *Proceedings of the 2008 ACM conference on Recommender systems*. ACM, 2008, pp. 267–274.
- [5] C. Wang, Y. Zheng, J. Jiang, and K. Ren, “Toward privacy-preserving personalized recommendation services,” *Engineering*, vol. 4, no. 1, pp. 21 – 28, 2018, cybersecurity. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S2095809917303855>
- [6] S. Badsha, X. Yi, and I. Khalil, “A practical privacy-preserving recommender system,” *Data Science and Engineering*, vol. 1, no. 3, pp. 161–177, Sep 2016. [Online]. Available: <https://doi.org/10.1007/s41019-016-0020-2>
- [7] V. Nikolaenko, S. Ioannidis, U. Weinsberg, M. Joye, N. Taft, and D. Boneh, “Privacy-preserving matrix factorization,” Nov 2013, pp. 801–812.
- [8] H. Kaur, N. Kumar, and S. Batra, “An efficient multi-party scheme for privacy preserving collaborative filtering for healthcare recommender system,” *Future Generation Computer Systems*, Mar 2018.
- [9] N. Youdao, “P4p: practical large-scale privacy-preserving distributed computation robust against malicious users,” *In Proc USENEX*, 2010.
- [10] D. Bachlechner, K. La Fors, and A. M. Sears, “The role of privacy-preserving technologies in the age of big data,” in *Proceedings of the 13th Pre-ICIS Workshop on Information Security and Privacy*, vol. 1, 2018.
- [11] G. Macintosh and J. W. Gentry, “Decision making in personal selling: Testing the kiss principle,” *Psychology & Marketing*, vol. 16, no. 5, pp. 393–408, 1999.

- [12] R. Ross, “Managing enterprise security risk with nist standards,” *Computer*, vol. 40, pp. 88–91, Aug 2007. [Online]. Available: [doi.ieeecomputersociety.org/10.1109/MC.2007.284](https://doi.org/10.1109/MC.2007.284)
- [13] J. Lu, D. Wu, M. Mao, W. Wang, and G. Zhang, “Recommender system application developments: A survey,” *Decision Support Systems*, vol. 74, Apr 2015.
- [14] D. Bokde, S. Girase, and D. Mukhopadhyay, “Matrix factorization model in collaborative filtering algorithms: A survey,” vol. 49, Apr 2015.
- [15] Y. Ar and E. Bostanci, “A genetic algorithm solution to the collaborative filtering problem,” *Expert Systems with Applications*, vol. 61, May 2016.
- [16] G. Takcs, I. Pilszy, B. Nmeth, and D. Tikk, “Matrix factorization and neighbor based algorithms for the netflix prize problem,” Jan 2008, pp. 267–274.
- [17] C. V. Yehuda Koren, Robert Bell, “Matrix factorization techniques for recommender systems,” *Computer*, vol. 42, no. 8, pp. 30–37, Aug 2009.
- [18] S. Liu, A. Liu, Z. Li, G. Liu, J. Xu, L. Zhao, and K. Zheng, “Privacy-preserving collaborative web services qos prediction via differential privacy,” Aug 2017, pp. 200–214.
- [19] D. Li, Q. Lv, H. Xia, L. Shang, T. Lu, and N. Gu, “Pistis: A privacy-preserving content recommender system for online social communities,” in *2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, vol. 1, Aug 2011, pp. 79–86.
- [20] D. Li, C. Chen, Q. Lv, L. Shang, Y. Zhao, T. Lu, and N. Gu, “An algorithm for efficient privacy-preserving item-based collaborative filtering,” *Future Generation Computer Systems*, vol. 55, Dec 2014.
- [21] V. Nikolaenko, S. Ioannidis, U. Weinsberg, M. Joye, N. Taft, and D. Boneh, “Privacy-preserving matrix factorization,” Nov 2013, pp. 801–812.
- [22] J.-Y. Jiang, C.-T. Li, and S.-D. Lin, “Towards a more reliable privacy-preserving recommender system,” *Information Sciences*, vol. 482, pp. 248 – 265, 2019. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0020025518310429>
- [23] A. Boutet, D. Frey, R. Guerraoui, A. Jégou, and A.-M. Kermarrec, “Privacy-preserving distributed collaborative filtering,” *Computing*, vol. 98, no. 8, pp. 827–846, Aug 2016. [Online]. Available: <https://doi.org/10.1007/s00607-015-0451-z>

- [24] J. Zhan, C. Hsieh, I. Wang, T. Hsu, C. Liao, and D. Wang, "Privacy-preserving collaborative recommender systems based on the scalar product," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 40, no. 4, pp. 472–476, Jul 2010.
- [25] M. Burkhart, M. Strasser, D. Many, and X. Dimitropoulos, "Sepia: Privacy-preserving aggregation of multi-domain network events and statistics," *Network*, vol. 1, no. 101101, 2010.
- [26] D. Li, X. Liao, T. Xiang, J. Wu, and J. Le, "Privacy-preserving self-serviced medical diagnosis scheme based on secure multi-party computation," *Computers Security*, vol. 90, p. 101701, 2020. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S016740481930238X>
- [27] H. Z. M. L. M Qiu, K. Gai, "Privacy - preserving smart data storage for financial industry in cloud computing," *Concurrency Computat Pract Exper*, vol. 30, Mar 2018. [Online]. Available: <https://doi.org/10.1002/cpe.4278>
- [28] V. S. Virupaksha and V. Bhramaramba, "Privacy preserving in banking sector," in *2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT)*, Jul 2016, pp. 571–575.
- [29] A. W.-C. F. P. S. Y. Benjamin C.M. Fung, Ke Wang, "Introduction to privacy-preserving data publishing: Concepts and techniques," *Chapman Hall/CR*, 2010.
- [30] A. Kiayias, B. Yener, and M. Yung, "Privacy-preserving information markets for computing statistical data," in *Financial Cryptography and Data Security*, ser. Lecture Notes in Computer Science, R. Dingledine and P. Golle, Eds. Springer Berlin Heidelberg, 2009, pp. 32–50.
- [31] Z. Ma, J. Ma, Y. Miao, K.-K. R. Choo, X. Liu, X. Wang, and T. Yang, "Pmkt: Privacy-preserving multi-party knowledge transfer for financial market forecasting," *Future Generation Computer Systems*, vol. 106, pp. 545 – 558, 2020. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0167739X19302596>
- [32] M. Kutylowski, Y. Wang, S. Xu, and L. Yang, "Special issue on social network security and privacy: Foreword to the special issue on social network security and privacy," *Concurrency and Computation: Practice and Experience*, p. e4414, Jan 2018.

- [33] Z. He, Z. Cai, and J. Yu, "Latent-data privacy preserving with customized data utility for social network data," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 1, pp. 665–673, Jan 2018.
- [34] A. Bielenberg, L. Helm, A. Gentilucci, and D. S. and, "The growth of diaspora - a decentralized online social network in the wild," in *2012 Proceedings IEEE INFOCOM Workshops*, Mar 2012, pp. 13–18.
- [35] C. A. Gomez-Urbe and N. Hunt, "The netflix recommender system: Algorithms, business value, and innovation," *ACM Trans. Management Inf. Syst.*, vol. 6, pp. 13:1–13:19, 2015.
- [36] B.-A. Yrekli, A. and C. Kaleli, "Exploring playlist titles for cold-start music recommendation: an effectiveness analysis," *Journal of Ambient Intelligence and Humanized Computing*, 2021.
- [37] Y. Z. liang Ya, "Design and implementation of medication recommending system for chronic hepatitis b," *Chinese Medical Equipment Journal*, vol. 38, no. 7, pp. 48 – 51, 2017.
- [38] C. Porcel, J. M. Moreno, and E. Herrera-Viedma, "A multi-disciplinar recommender system to advice research resources in university digital libraries," *Expert Syst. Appl.*, vol. 36, no. 10, pp. 12 520–12 528, Dec. 2009. [Online]. Available: <http://dx.doi.org/10.1016/j.eswa.2009.04.038>
- [39] C. Porcel and E. Herrera-Viedma, "Dealing with incomplete information in a fuzzy linguistic recommender system to disseminate information in university digital libraries," *Knowl.-Based Syst.*, vol. 23, Sep 2009.
- [40] H. Xie, Y. R. Yang, A. Krishnamurthy, Y. G. Liu, and A. Silberschatz, "P4p: Provider portal for applications," in *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4. ACM, 2008, pp. 351–362.
- [41] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.
- [42] N. Balachandran and S. Sanyal, "A review of techniques to mitigate sybil attacks," *arXiv preprint arXiv:1207.2617*, 2012.
- [43] P. T.P., *Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing*. Springer, Berlin, Heidelberg, 1992, vol. 576.
- [44] P. Feldman, "A practical scheme for non-interactive verifiable secret sharing," in *28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*. IEEE, 1987, pp. 427–438.



- [45] M. L. Baldoni R., Querzoni L., “Italian cybersecurity report - controlli essenziali di cybersecurity,” *Future Generation Computer Systems*, 2017.
- [46] P. M. C. Crovini, G. Ossola, “Cyber risk. the new enemy for risk management in the age of globalisation,” *Management Control*, 2018.
- [47] S. G. Feringa A., Goguen A., “Risk management guide for information technology systems nist special publication 800 30,” 2002.
- [48] E. C. E. Biffis, “Satellite data and machine learning for weather risk management and food security,” *Risk Anal*, vol. 37, no. 8, pp. 1508–1521, Aug 2017.
- [49] U. Makov and J. Weiss, *Predictive Modeling for Usage-Based Auto Insurance*, ser. International Series on Actuarial Science. Cambridge University Press, 2016, vol. 2, p. 290308.
- [50] G. W. P., “Statistical machine learning and data analytic methods for risk and insurance.” no. 8, 2017.
- [51] D. M.-P. S. G. W. Peters, R. Cohen, “Understanding cyber risk and cyber insurance,” *Macquarie University Faculty of Business and Economics Research Paper*, Jan 2018.
- [52] C. Biener, M. Eling, and J. H. Wirfs, “Insurability of cyber risk: An empirical analysis,” *Geneva Papers on Risk and Insurance: Issues and Practice*, vol. 40, no. 1, pp. 131–158, Jan 2015. [Online]. Available: <https://www.alexandria.unisg.ch/238242/>
- [53] T. A. A. K. B. Geluvara, P. M. Satwik, “The future of cybersecurity: Major role of artificial intelligence, machine learning, and deep learning in cyberspace,” in *International Conference on Computer Networks and Communication Technologies*, Jul 2017.
- [54] D. Polemi, T. Ntouskas, E. Georgakakis, C. Douligeris, M. Theoharidou, and D. Gritzalis, “S-port: Collaborative security management of port information systems,” in *IISA 2013*, Jul 2013, pp. 1–6.
- [55] D. Friday, S. Ryan, R. Sridharan, and D. Collins, “Collaborative risk management: a systematic literature review,” *International Journal of Physical Distribution & Logistics Management*, vol. 48, no. 3, pp. 231–253, 2018. [Online]. Available: <https://doi.org/10.1108/IJPDLM-01-2017-0035>

- [56] D. K. Tosh, S. Shetty, S. Sengupta, J. P. Kesan, and C. A. Kamhoua, "Risk management using cyber-threat information sharing and cyber-insurance," in *Game Theory for Networks*, L. Duan, A. Sanjab, H. Li, X. Chen, D. Materassi, and R. Elazouzi, Eds. Cham: Springer International Publishing, 2017, pp. 154–164.
- [57] G. Settanni, F. Skopik, Y. Shovgenya, R. Fiedler, M. Carolan, D. Conroy, K. Boettinger, M. Gall, G. Brost, C. Ponchel, M. Haustein, H. Kaufmann, K. Theuerkauf, and P. Olli, "A collaborative cyber incident management system for european interconnected critical infrastructures," *Journal of Information Security and Applications*, vol. 34, Jun 2016.
- [58] R. F. F. Skopik, G. Settanni, "The importance of information sharing and its numerous dimensions to circumvent incidents and mitigate cyber threats," *Collaborative Cyber Threat Intelligence*, vol. 4, 2017.
- [59] *OpenVAS - Open Vulnerability Assessment System*. [Online]. Available: <http://openvas.org/>