Grant Agreement No.: 611001

UCN

Instrument: Collaborative Project

Call Identifier: FP7-ICT-2013-10

Objective 1.6: Connected and Social Media

D4.2: Specification and preliminary design of privacy preserving mechanisms for data management

Due date of deliverable: 31.03.2015

Actual submission date: 31.03.2015

Start date of project: October 1st 2013 Duration: 36 months

Project Manager: Henk Heijnen, Technicolor SA

Author(s): EURECOM, NICTA, Portugal Telecom

Editor: EURECOM

Revision: v.1.0

Abstract

This deliverable describes a number of privacy preserving primitives for data collection and data processing. Thanks to these solutions, end-users will be able to control their privacy exposures to a certain degree and make decisions on how third parties, such as recommenders, can exploit their data. The proposed mechanisms are designed under two security models, namely, *partial access* and *full access* models which were defined in the previous deliverable of this work package.

Project co-funded by the European Commission in the 7th Framework Programme (2007-2013)		
Dissemination Level		
PU	**Public**	✓
PP	**Restricted to other programme participants (including the Commission Services)**	
RE	**Restricted to a group specified by the consortium (including the Commission Services)**	
CO	**Confidential, only for members of the consortium (including the Commission Services)**	

TABLE OF CONTENTS

1 INTRODUCTION

The UCN (User-Centric Networking) project aims at collecting and exploiting as much information as possible about end-users in order to further provide them a high quality and personalized service in return. The recent technology developments allow millions of people to collect and share data on a massive scale. Such data which is defined as the "user context" in WP2, allow third parties such as recommenders to leverage rich knowledge about end-users.

In D2.1 [1], authors discuss that data can be collected from multiple heterogeneous devices (smartphones, sensors, gateways), and range from basic network statistics to more sophisticated and detailed information about end-user activities (door/window state, arrival/departure time, social network information). While this new data collection paradigm opens up big opportunities for the market, the exploding amount of data which can be highly sensitive, raises serious privacy concerns which the user may even not realize.

The aim of this deliverable is to propose a number of privacy preserving primitives both for the data collection and the further data processing phases, in order to allow end-users to have control on their privacy exposures. In D4.1 [2], we identified three different security models based on our investigation on users' understandings of privacy and the use-cases explored within WP5 [3]. The requirements of these models depend on the end-user's privacy level preferences. While the first model does not allow any access to users' data, the remaining ones assume that users are willing to share some information.

The privacy preserving solutions we propose in this deliverable, are designed under these two models, namely partial access model and full access model. In the partial access model, whereby third parties are allowed to derive some information about users' data while not being able to have full access, we propose two different mechanisms: the first solution uses secure multi-party computation to achieve privacy preserving aggregation during the collection phase; the second mechanism ensures data obfuscation for recommendation systems using differential privacy. The third security model assumes that an end-user accepts to reveal its data to authorized third parties while still being able to keep the control over this access. The first scheme describing a privacy preserving recommendation solution based on matrix factorization and designed under this third security model requires the user to reveal some data only when it is needed and this data is immediately destroyed by the third party (here, the recommender) after its use. We further propose a preliminary design of privacy preserving lookup solution which ensures that only authorized parties can search for some words (chosen by the end-user) in a privacy preserving and efficient manner. Finally, an initial version of a sticky policy framework is proposed to control the access to data based on its dedicated policy.

2 UCN PRIVACY MODELS

In this section, we briefly discuss the privacy challenges of user-centric networks and provide a quick overview of the use-cases presented in D5.1 [3] and the privacy models investigated in D4.1 [2].

2.1 UCN Privacy Challenges

The UCN framework relies on devices called "Personal Information Hub" which communicate with the smart home appliances while hooked-up to users' routers. As such, PIH relays information from the appliances at home to the outside, for instance, if an intrusion is detected, the PIH informs the user via an SMS. Since PIH has full access to sensitive information, users require control on how the PIH communicates with the outside world. Namely, users wish to be able to change the configuration of PIH so as to determine which information third parties can fetch. Indeed, the main purpose of PIH is to collect information from various sensors and devices present in the house in the aim of deriving and inferring some meaningful information about PIH users. The derived information can then be either leveraged by users to detect an intrusion or faulty appliances, or exploited by third party service providers to improve users' experience by generating relevant recommendations or targeted advertisements.

In this environment, privacy concerns arise when third parties access the data collected by the PIH. This access is actually inevitable for two reasons: First, the PIH is a constrained device with stringent storage capabilities, which means that in order for the user to keep his data, he has to outsource it to a cloud server. Second from a commercial perspective, the main point of installing PIH is to enhance third party services such that they become more personalized and pertinent. Seen in this light, PIH hinders users' privacy, particularly, if adequate security counter-measures are not installed. An example of such counter-measures is "encryption": Data will be encrypted before leaving the PIH. The challenge however, is how third parties will process the encrypted data generated by individual users without actually decrypting it. Moreover, if decryption is mandatory, then it is important to limit its impact on users' privacy by either encouraging third parties to discard the decrypted data when it is no longer needed, or by making sure that the decrypted information is not traced back to individual users.

To illustrate how PIH is useful to various services and applications, D5.1 [3] elicited a number of use-cases from UCN partners. These use-cases were then elided into two main categories in D4.1 [2] that we recall in the following section:

2.2 UCN Use-cases

Based on their application field, use-cases in D5.1 [3] were classified into two categories:

2.2.1 Recommendations

This category subsumes use-cases that aim at collecting (personal) information from users (i.e. PIH), in order to build behavioural profiles that can later be used by recommenders interested in predicting and anticipating users' needs. An illustrative example of this category is TV recommendations investigated in D5.1 [3]. In this example, each PIH constructs and refines a profile based on the viewing habits of its owner. The PIH in this example shares a sanitized version of the user's profile with interested third parties (i.e. recommenders). Recommenders use the information received from different PIHs in the system to compute and then send relevant TV shows' recommendations to PIH users. Since PIH shares information with recommenders, it is very important to limit the amount of information the recommender learns about the user. Ideally, the recommender should only learn the information that is vital to the correctness of its computations (i.e. the PIH should ensure/enforce minimal information disclosure to third parties).

2.2.2 Smart Homes

The use-cases in this category deal with applications that orchestrate the interactions between different smart devices at home (for instance, the interaction between a motion detector and a user's smart phone). As these applications generate huge amount of data, they give rise to some serious security and privacy threats. The PIH by construction does not accommodate the storage of huge amount of data, and as a result, users are bound to either discard the data stored at the PIH or outsource it to a cloud server. Outsourcing data seems to be the most viable option: For example, a user may want to keep the video-streams generated by its surveillance system for a few months for inspection. Due to the sensitive nature of users' data, the PIH will encrypt data before outsourcing, yet there should still be a means to process this data without downloading it.

In light of these use-cases, D4.1 [2] introduced dedicated security models that together define a flexible security framework that can be tuned to users' security needs without sacrificing utility.

2.3 UCN Privacy Models

In D4.1 [2], we identified three security models (cf. Figure 1) that cover most scenarios that we encounter in real life. Going from the most conservative security model to the more flexible ones, we find:

2.3.1 No Access

In this model, users are not required to share any identifiable personal information with third parties. This may be achieved by having a "Broker" that relays encrypted information from users to authorized third parties. From this encrypted information, third parties get some insights on users in the system, but thanks to the broker in the middle, they cannot learn which users are contributing to the data collection process, and for that matter cannot link the inferred insights to individual users. This model is well suited for targeted advertisement

applications, wherein users are interested in receiving advertisements about relevant products, but they want to neither reveal their personal interests to the broker nor reveal their identities to the advertiser. Solutions based private matching like [4] can be considered as potential candidates for this model.

2.3.2 Partial Access

As the name implies, users are willing to share some information with third parties. This security model covers scenarios where the shared information is not personal information; rather it is aggregate information that, for instance, describes the behaviour of a group of users. One approach to meet the privacy requirements of this model is to combine secret sharing techniques and homomorphic encryption, in such a way that third parties learn only the aggregate value of users' information and nothing else. Although the aggregate information is not personal information per se, it can be combined with some side-channel information to undermine the privacy of individual users. Therefore, in this model, we also take into account the case where individual users share perturbed information (i.e. noisy users' information) with third parties. The third parties use the shared information albeit perturbed to derive statistical information (that is not 100% exact) about a population of users. The amount of noise added to the users' information is calibrated according to the differential privacy framework. This approach allows third parties to obtain reasonably accurate statistical information, while masking users' individual inputs. Two different privacy solutions targeting this security model are proposed and described in detail in section 3.

2.3.3 Full Access

Here users are more inclined into allowing third parties to access a part of their information. They still want the access to be controlled (i.e. users know exactly what information an authorized party accesses) and to be revoked or granted at any point in time. As discussed in the previous deliverable, such a model comes in handy in scenarios whereby users outsource their data to a cloud server and still would like to allow third parties (recommenders for instance) to search the outsourced data. The users however want to neither divulge the content of their data to the cloud servers nor allow third parties to learn more than the search result (i.e. whether a keyword is present in the users' data). This model also considers the case where a third party decrypts users' data before processing. Generally, recommenders use dedicated algorithms that often do not have privacy preserving variants (i.e. variants that operate on encrypted data), and if they do, their privacy preserving variants are computationally demanding and might hamper the performances of the recommendation system. While in such a case decryption is inevitable, it is important to ensure that recommenders do not retain data indefinitely, for instance, by encouraging recommenders to use trustworthy computing platforms. Section 4 provides some new techniques that are secure under this *full access* model.

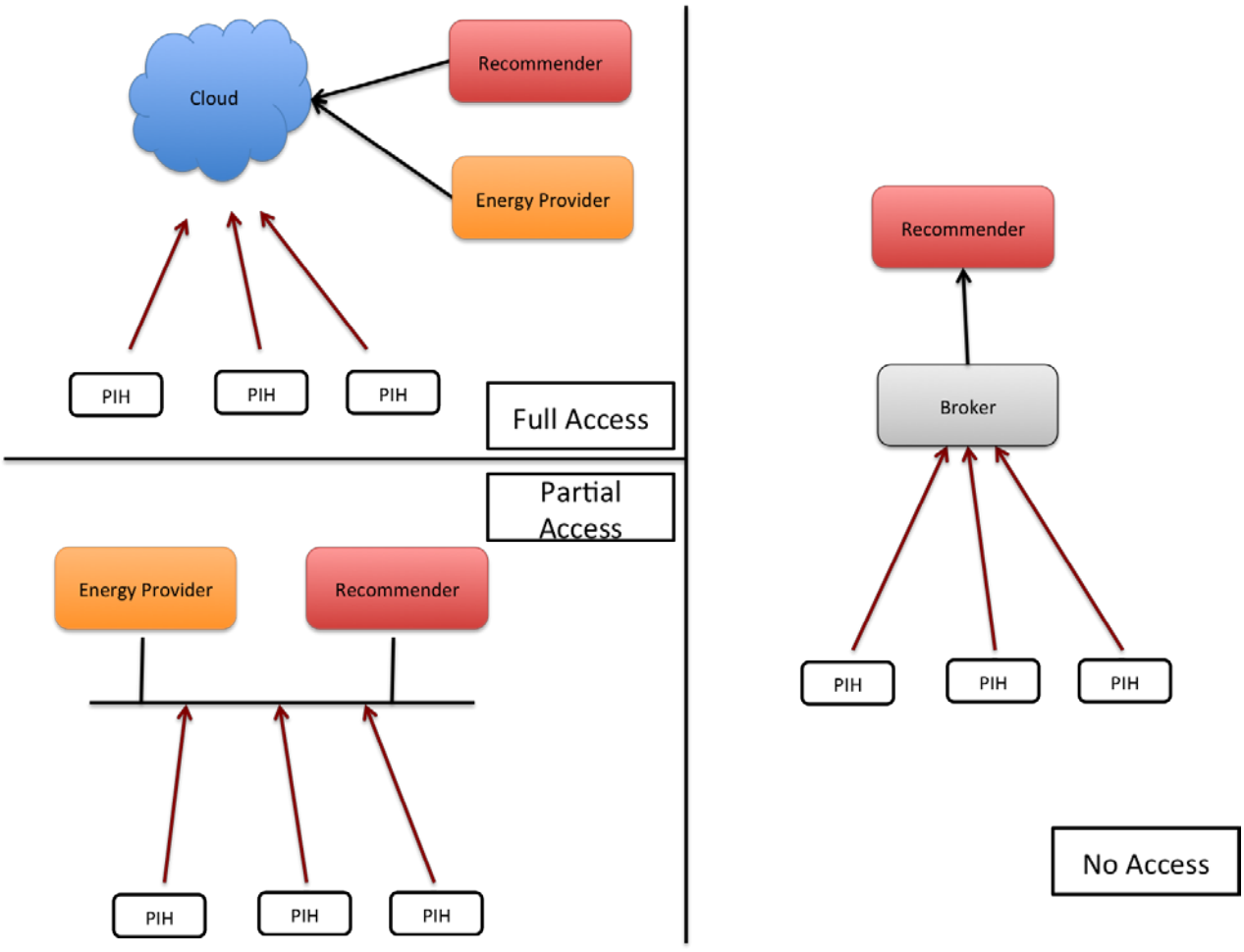


Figure 1 UCN Security Models

3 PRIVACY-PRESERVING SOLUTIONS FOR THE PARTIAL ACCESS MODEL

Here, we introduce two schemes that tailor well-established cryptographic concepts to ensure users' privacy. The first scheme tackles the issue of privacy preserving data aggregation for dynamic groups of users by means of *secure multi-party computation*; whereas the second solution addresses the privacy challenges pertaining to recommender systems via data obfuscation and *differential privacy*.

3.1 Privacy-preserving Data Aggregation

In UCN, we consider a scenario where an aggregator aims to compute the sum of the private data of a set of users without learning users' individual input. In accordance with the work of [5] and [6], we only focus here on time-series data which is a series of data point measurements that are observed at equally spaced time-intervals. A naive approach to compute the aggregate is to encrypt each user's individual inputs using the public key of the aggregator. This solution however relies on a fully trusted aggregator that learns the users' individual data before aggregation. To counter this shortcoming, the work in [5] and [6] combines secret sharing and additively homomorphic encryption to enable the aggregator to compute the sum of users' data without compromising users' privacy. The idea is to have a trusted third-party called key dealer that provides each user in the system with a secret share, while supplying the aggregator with a secret key computed as the sum of users' shares.
Each user encrypts his private data using his secret share and forwards the resulting ciphertext to the aggregator. The aggregator in turn combines the received ciphertext in such a way it obtains an encryption of the sum of users' input that can be decrypted using the aggregator secret key (i.e. the sum of users' secret shares).
Although such solutions prevent the aggregator from getting users' private inputs, they suffer from two limitations. The first is that they assume a fully trusted dealer that does not have any incentive to jeopardize users' privacy. The second limitation is that these solutions only support static groups of users and therefore, they are fault intolerant. Namely, in the case of user failures the aggregator can no longer compute the aggregate sum. We propose thus a solution for privacy-preserving data aggregation of time-series data that draw upon the work of [6] to support dynamic group management and arbitrary user failures. The idea is that rather than involving a key dealer to generate and distribute the secret shares to the users in the system, we rely on a semi-trusted entity that we call collector, which acts as an intermediary between the users and the aggregator. This collector helps the aggregator compute the aggregate sum of users' individual data without any prior key distribution by a trusted key-dealer.

3.1.1 Idea of Solution

The homomorphic scheme suggested by Joye and Libert [6] allows an untrusted aggregator to compute the sum without disclosing the individual data of users in the system. Nevertheless,

to implement this functionality, a fully trusted dealer generates and distributes secret shares to users in the system. As a result, the dealer can decrypt the users' cipheretexts and learn the cleartext value of the users' inputs. The solution we propose improves the scheme of Joye and Libert in the following aspects:

- ➢ It does not require a key dealer to distribute secret shares to users.
- ➢ It supports dynamic groups. Contrary to the Joye-Libert scheme, join and leave operations incur no computation or communication overhead at the users' side.
- ➢ It is resilient to arbitrary user failures that may occur due to communication errors or hardware failures.

To concretely implement our solution, we employ two techniques:

– *Responsibility splitting mechanism:* For each time interval t, each user U_i sends an encryption of his private data sample to the aggregator and a partial decryption key $dk_{i,t}$ to the semi trusted collector, in such a way that neither the aggregator nor the collector can violate the privacy of individual data provided by users.

– *Self-generation of secret keys:* Each user U_i chooses his secret key $sk_{i,}$ independently from other users. The aggregator on the other hand, generates a random secret key sk_A and for each time interval t, it publishes an obfuscated version $pk_{A,t}$ of sk_A.
Each user U_i accordingly encrypts his private data input $x_{i,t}$ with his secret key sk_i using the Joye-Libert cryptosystem, and sends the computed ciphertext $c_{i,t}$ to the aggregator. User U_i also randomizes his secret key sk_i using $pk_{A,t}$ and sends the resulting partial decryption key $dk_{i,t}$ to the collector through a secure channel. The collector in turn computes a function of the partial decryption keys it has received and forwards the output dk_t to the aggregator. Upon receiving the ciphertexts $c_{i,t}$ and the partial decryption key dk_t, the aggregator uses its secret key sk_A to decrypt the sum of $x_{i,t}$ for time interval t.

Before presenting our solution for privacy-preserving aggregation, we briefly describe the Joye-Libert scheme.

3.1.2 Joye-Libert Scheme

The solution of Joye and Libert consists of three phases:

- ➢ *Setup*: A trusted dealer randomly selects two safe primes p and q and sets $N = pq$. It then defines a cryptographic hash function $H: \{0,1\}^* \rightarrow Z_{N^2}$ and outputs the public parameters (N, H). Eventually, the dealer gives each user U_i a secret share sk_i in $[0, N^2]$ and sends $sk_A = -\sum sk_i$ to the untrusted aggregator.
  Henceforth all computations are done "$mod\ N^2$" unless mentioned otherwise.
- ➢ *Encrypt*: For each time interval t, each user U_i encrypts his private input $x_{i,t}$ using the secret key sk_i and outputs ciphertext $c_{i,t} = (1 + x_{i,t}N)H(t)^{sk_i}$.

- *Aggregate*: On receiving $c_{i,t}$, the untrusted aggregator computes:

$$P_t = \prod c_{i,t} = \left(1 + \sum x_{i,t} N\right) H(t)^{\Sigma sk_i} = \left(1 + \sum x_{i,t} N\right) H(t)^{-sk_A} \, mod \, N^2$$

Then, it recovers the sum of $x_{i,t}$ by computing in Z:

$$\sum x_{i,t} = \frac{P_t \, H(t)^{sk_A} \, mod \, N^2}{N}$$

3.1.3 Description of Solution

Our protocol for privacy-preserving data aggregation involves four phases:

- *Setup*: A trusted third party selects two safe primes p and q, sets $N = pq$ and selects a cryptographic hash function $H: \{0,1\}^* \rightarrow Z_{N^2}$. Later, the trusted third party publishes (N, H) and goes offline. Next, the aggregator generates a random secret key sk_A in Z_N^*, whereas each user U_i independently chooses his random secret key sk_i in $[0, N^2]$.
- *Encrypt*: For each time interval t, each user U_i encrypts his private input $x_{i,t}$ using the secret key sk_i and the Joye-Libert scheme. Afterwards, user U_i sends the resulting ciphertext $c_{i,t} = (1 + x_{i,t}N)H(t)^{sk_i}$ to the aggregator.
- *Collect*: In each time interval t, the aggregator advertises the value $pk_{A,t} = H(t)^{sk_A}$. Each user on the other hand computes the partial decryption key $dk_{i,t} = H(t)^{sk_A sk_i}$ and sends $dk_{i,t}$ to the collector through a secure channel. Upon receipt of $dk_{i,t}$, the collector computes the partial decryption key : $dk_t = \prod dk_{i,t} = H(t)^{sk_A \Sigma sk_i}$ and sends the result to the aggregator.
- *Aggregate*: On receiving ciphertexts $c_{i,t}$ and partial decryption key dk_t, the aggregator computes:

$$P_t = \left(\prod c_{i,t}\right)^{sk_A} = \left(1 + sk_A \sum x_{i,t} N\right) H(t)^{sk_A \Sigma sk_i} \, mod \, N^2$$

Then it computes in Z the sum of $x_{i,t}$ as:

$$\sum x_{i,t} = \frac{\frac{\left(\frac{P_t}{dk_t} - 1\right)}{sk_A} \, mod \, N^2}{N}$$

For a more thorough description of the solution, the reader may refer to the paper [7].

3.1.4 Integration within UCN

The solution described above can be easily integrated within the UCN framework. For instance, sensors gauging energy consumption will relay their measurements to the PIH. The PIH in turn is synchronized with both the energy provider (i.e. the aggregator) and the network carrier (i.e. the collector). In each time interval, the PIH first computes the average user consumption using the data collected from the sensors, then encrypts the average and computes a partial decryption key, and finally sends the resulting ciphertext and partial decryption key to the energy provider and the network carrier respectively. This implies that

PIHs in UCN need dedicated cryptography modules to implement an algorithm for random key generation and the Joye-Libert encryption depicted in Section 3.1.3 (cf. Figure 2).

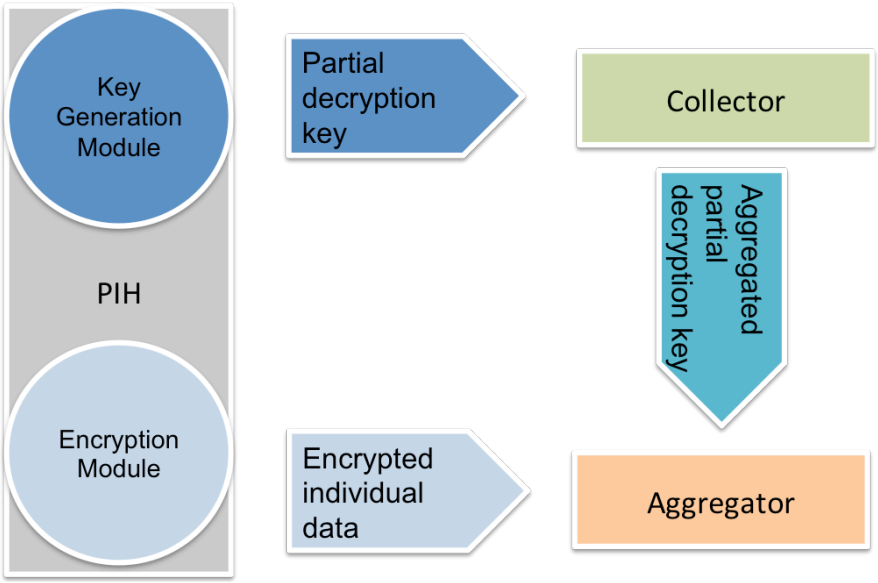


Figure 2 Privacy-preserving Aggregation within UCN

3.2 Recommendations with Input Obfuscation

One of the ways to protect user privacy against either an untrusted service provider or third-party inference of users' private information is to obfuscate the users' inputs. The obfuscation is performed by a user agent (e.g., software running on the PIH on behalf of the user). Client side data obfuscation has been studied extensively in the context of data mining [8] [9] and it has the advantage of providing privacy protection without requiring the users to trust the online system, but in turn it reduces the accuracy of the recommendations. Contrary to the majority of previous work, which targets a specific level of privacy and aims to maximize the utility (i.e., accuracy of the recommender system) within the predefined privacy constraints, the solution we design for UCN considers a user-acceptable trade-off between utility and privacy and offers a suitable mechanism to reach this trade-off.

3.2.1 System Architecture

We consider a generic system architecture, where the users access an online service that also includes recommender system functionality (within the system, or provided externally, by an analytics service), as shown in Figure 3.

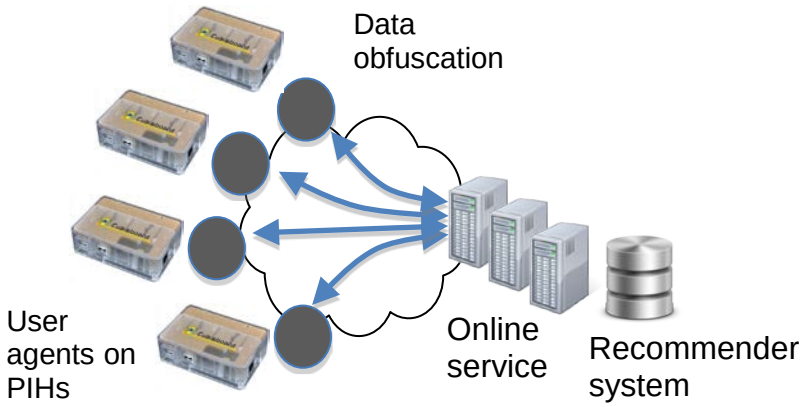


Figure 3 System Architecture

Users utilize the online (shopping or content provider) service and also use the associated recommendation system. For the latter, they provide ratings for the items purchased or viewed using a client-side agent installed on the PIH. The software obfuscates the users' ratings before they are sent to the recommender system by adding noise, where the magnitude of noise is calibrated according to the level of privacy chosen by the user. We note that the privacy level needs to be determined by considering two contradicting objectives: users' privacy preferences, and system utility (receiving meaningful recommendations on relevant products).

The recommender system acquires rating and transaction data from users and generates personalized recommendations. The objective of the recommender system is to accurately predict user preferences for future viewings or purchases. For our study, we utilize *matrix factorization* **[10]**, the state-of-the-art technique in recommender systems. The input to the recommendation process is the set of user ratings. The matrix factorization process derives two low-rank matrices, one that captures user profiles, and one that captures item profiles. The matrices are constructed such that they provide an accurate approximation for the known ratings, and are therefore expected to make fairly accurate predictions also for unknown ratings.

3.2.2 Measuring and Applying Privacy Protection

To measure the privacy protection resulting from data obfuscation (perturbation of the recommender system input), we rely on the differential privacy framework **[11]**. Differential privacy sets limits on the influence that any particular record in the input could have on the outcome of a computation. Differential privacy guarantees that an adversary will not be able to infer any particular rating from the outcome of the computation regardless of the computational power or the background knowledge available to the adversary. A parameter ε controls the level of privacy, where smaller values of ε provide stricter bounds on the influence of any particular input record on the outcome, and therefore provide better privacy.

The Laplace mechanism **[11]** can be used to perturb the user response while conforming to ε-differential privacy. A Laplace distribution with a scale parameter $\boldsymbol{b}$ (and variance $\boldsymbol{\sigma^2 = 2b^2}$) has probability density function $\mathbf{Pr}(\boldsymbol{x}|\boldsymbol{b}) = \frac{\mathbf{1}}{\mathbf{2}\boldsymbol{b}}\mathbf{exp}(-\frac{|\boldsymbol{x}|}{\boldsymbol{b}})$. Given the range $\boldsymbol{\Delta}$ of the user's possible answers, the Laplace mechanism obtains ε-differential privacy by adding to each input noise sampled from the Laplace distribution, with its scale $\boldsymbol{b}$ calibrated through the relation: $\boldsymbol{\varepsilon = \frac{\Delta}{b}}$.

Example: Consider a 5-point rating scale commonly used in movie ratings, with ratings between 1 (very poor) and 5 (excellent). The privacy options considered by users should be simple enough for the lay user to understand, so we adopt a simple set of four privacy levels: *no*, *low*, *medium* and *high*. We use noise with standard deviation $\boldsymbol{\sigma = 0}$ for no privacy, $\boldsymbol{\sigma = \sqrt{2}}$ for low privacy, $\boldsymbol{\sigma = 4}$ for medium privacy, and $\boldsymbol{\sigma = 4\sqrt{2}}$ for high privacy (note that the obfuscated responses will consequently be real-valued rather than integers). With $\boldsymbol{\Delta = 5 - 1 = 4}$, the *low*, *medium* and *high* privacy settings correspond to $\boldsymbol{\varepsilon = 4, \varepsilon = 1.4}$ and $\boldsymbol{\varepsilon = 1}$ respectively.

3.2.3 An Adaptive Obfuscation Mechanism

Since getting adequate prediction accuracy is a key requirement of the system, we propose a practical approach to balance privacy and accuracy in this context. The goal is to allow users to receive reasonably accurate recommendations, but without completely compromising their privacy. To this end, we propose *PrivacyCanary*, an interactive recommendation service, which can be probed by the users to assess the expected accuracy of the predicted ratings. This feedback mechanism allows the users to adapt obfuscation according to the current performance of the system.

In this approach, users obtain predictions from the recommender system and have a pre-determined range of acceptable prediction accuracy. To control the accuracy of received recommendations and the level of privacy loss, each user agent utilizes a reference set of *canary items*. The canary set is a set of items that the user rated without disclosing the ratings to the recommender system. By probing the recommender engine to obtain the predictions for these canary items, the user agent can estimate the current recommendation accuracy and select the appropriate level of obfuscation to maintain accuracy within the acceptable range. Essentially, the user agent adapts the obfuscation level of subsequently submitted ratings according to the expected accuracy, so privacy protection can be maximized while maintaining reasonable accuracy in future predictions.

The following protocol workflow is repeated each time a user rates a new item:

1. The user probes the system: the user keeps a set of item ratings as canary set and probes the system for rating predictions for the items in the set.
2. The recommender system sends the predictions for the canary items: the recommender system executes the recommendation algorithm to derive predictions for the canary set.
3. The user adjusts the obfuscation level and reveals the obfuscated rating for a new item: the exact process, which takes into account the targeted accuracy level and modifies the obfuscation level according to the accuracy bounds, is described below.

The basic premise of this solution is that, rather than selecting a privacy level and evaluating the resulting accuracy, we select a desired accuracy level and adjust privacy (i.e., the level of added noise) accordingly. We assume that each user has a set of item ratings that are not disclosed to the recommender system, and refer to them as the canary items. Let $\boldsymbol{I_U}$ denote the set of canary items for user $\boldsymbol{U}$, and let $\boldsymbol{r_{U,i}}$ be the rating that user $\boldsymbol{U}$ gave to canary item i. We denote by $\boldsymbol{r'_{U,i}}$ the system prediction for $\boldsymbol{U}$'s rating for item *i*. The Root Mean Square Error (RMSE) of user $\boldsymbol{U}$'s canary set is given by: $\boldsymbol{RMSE_{I_U}} = \sqrt{\sum_{i \in I_u} \left(r_{U,i} - r'_{U,i}\right)^2 / |I_U|}$. We rely on the RMSE of the canary set as a measure for the expected accuracy of the recommendations.

To control the quality of information that the recommender system has about him, the user aims to maintain recommendation system accuracy while obfuscating his ratings (sent sequentially to the system). The user defines the lower and upper bounds on prediction accuracy (lower error will result in accurate profiling of the user, while higher error will result in meaningless recommendations), denoted by $\boldsymbol{e_l}$ and $\boldsymbol{e_u}$ respectively, and consequently adjusts the obfuscation level if the prediction errors are not in the desired region – the level of obfuscation is increased when the expected RMSE is below $\boldsymbol{e_l}$, and decreased when it is above $\boldsymbol{e_u}$. Since the obfuscation is applied only to the user ratings, our approach can be used with any recommendation system.

3.2.4 Evaluation

Full evaluation of the proposed approach was conducted in [12]. We show below two main outcomes that demonstrate the advantages of the adaptive algorithm. The results rely on the MovieLens 100k dataset,[1] and use the Matrix Factorization algorithm supplied in the MyMediaLite[2] recommender system library. The canary set for each user is composed of 10 random movies from the user's profile. The main evaluation criterion is the prediction error, based on the average root mean square error (RMSE) between the predicted ratings and actual ratings for each user, using a "leave-one-out" cross validation approach. We assume a target accuracy values of $e_l = 2$ and $e_u = 2.1$, i.e., that users will be willing to sacrifice privacy to bring the RMSE below 2.1, but would like to introduce more noise once the accuracy drops below 2.

Figure 4 shows the average error in RMSE over an evaluation (hold-out) set over time (as the users rate more movies) for various (fixed) obfuscation levels and for the adaptive approach. Each point in the plot is obtained by averaging the RMSE over 100 simulation runs per user, and then averaging it over 100 random users selected for evaluation. Fixed-level obfuscation imposes a penalty, increasing the error in prediction. However, the impact of obfuscation on prediction error fades as the users keep using the same fixed obfuscation level. In contrast, the adaptive obfuscation algorithm effectively keeps the average prediction error stable as users rate more movies, within the accuracy bounds set by the user (i.e., on average the prediction error is kept at about the same level).

[1] http://www.grouplens.org/node/73
[2] http://mymedialite.net/

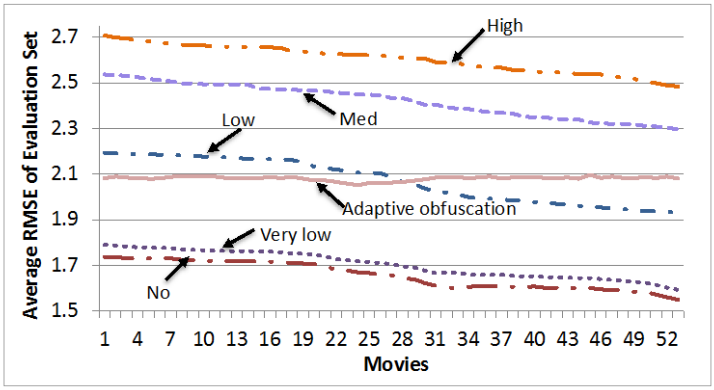


Figure 4 Prediction accuracy for an evaluation set based on a randomly selected canary set

We note that the resulting privacy provided by the system is best effort, as each individual rating item belonging to a selected user will be protected according to the corresponding level of obfuscation noise.

A second factor that the adaptive mechanism addresses is the dependence on other users' behavior. We consider a target user $\boldsymbol{U}$, who adaptively obfuscates his ratings based on a randomly chosen canary set and explore the scenario where a fraction of other users obfuscate their responses with a fixed privacy level $\boldsymbol{\varepsilon}$. We evaluate the impact of the obfuscation conducted by other users on the prediction accuracy of the user $\boldsymbol{U}$, to see whether a user can effectively control his prediction accuracy by probing the system and changing the obfuscation level based on the feedback, regardless of other users conducting obfuscation in the system.

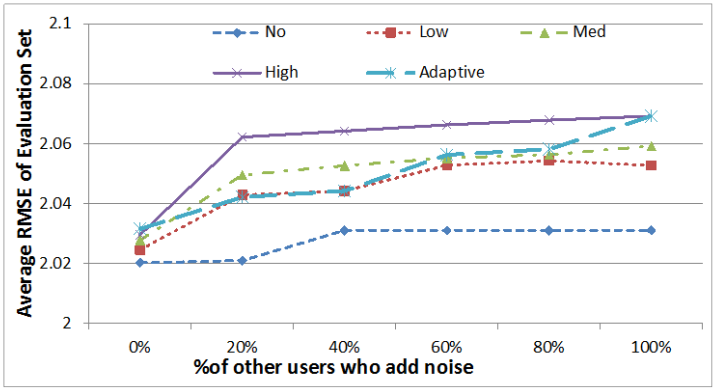


Figure 5 Accuracy of the predictions in an evaluation set, when % of other users obfuscate their ratings

Figure 5 shows the RMSE in predictions for the evaluation set of randomly selected 100 users, with the same adaptive accuracy bounds as in the previous experiment. Each line in the plot depicts the average RMSE in predictions of a user who adaptively obfuscates ratings while a fraction of the other users in the system choose *no*, *low*, *medium* and *high* fixed-level

privacy. Each point in the plot is obtained by averaging RMSE over 100 simulation runs per user and then averaging it over 100 random users selected for evaluation. We observe that when a user adaptively obfuscates his ratings, his accuracy in predictions of the evaluation set stays stable over time, despite the obfuscation conducted by other users. Adaptive obfuscation of user ratings enables the user to keep the accuracy fixed between the desired bounds regardless of the fraction of other users who obfuscate their ratings according to a fixed level of privacy. As evident in Figure 5, this is also the case when other users apply the adaptive obfuscation mechanism.

4 PRIVACY-PRESERVING SOLUTIONS FOR THE FULL ACCESS SECURITY MODEL

In this section, we relax the privacy restrictions a little bit more and define solutions whereby third parties can have full access over end-users' data while the user can still have some control over the access rights and for example revoke a third party whenever desired.

4.1 Recommendations without User Data Retention

While encryption technologies mitigate the risk of eavesdropping when data is transferred to the recommender system, the retention of the data by the recommender exposes the users to additional privacy risks. Even seemingly innocuous preference information, like that typically stored by recommender systems, introduces privacy risks, as demonstrated in scenarios such as 1) inference of undisclosed sensitive personal information, e.g., relationship status or sexual orientation, through access to public user data **[13] [14] [15] [16]**; 2) trading or brokering of personal data to an untrusted third party;[3] or 3) disclosure of sensitive personal information to law enforcement agencies, even without a warrant **[17]**. These issues fuel a growing concern that motivates keeping personal user data away from a centralized storage, thereby mitigating data disclosure risks.

In this section, we describe an approach that leverages Matrix Factorization (MF) techniques **[10]** to investigate the use of a *semi-decentralized recommender*, which retains neither the user ratings nor the latent user vectors. Instead, user ratings are stored on the user's side, e.g., on the PIH. When a user rates a new item or a recommendation is required, the user's ratings are provided to the MF recommender, which derives the latent user vector and updates the latent item matrix. Then, item ratings are predicted and the recommendations are generated. When the interaction with the user is concluded, the recommender discards both the ratings and the latent user vector. This semi-decentralized setting mitigates the above risks by not retaining permanently any user-specific data that could be exploited for the inference of sensitive personal information, while allowing service providers to retain content-specific data, crucial for the provision of recommendations.

4.1.1 Recommendations with Matrix Factorization

Consider a set of *n* users who assign ratings to a set I of *m* items. We denote by S the set of available ratings, where each element is a triplet $(U, i, r_{U,i})$, and $r_{U,i}$ denotes the rating that user U assigned to item i. We denote by $S_V = \{(U, i, r_{U,i}) \in S | U = V\}$ the set of known ratings of user V. The recommendations are generated by predicting the values of unknown ratings. MF achieves this by learning two low-rank matrices with d latent factors: $P_{n\times d}$ for users and $Q_{m\times d}$ for items, where a row p_U in P pertains to user U and a row q_i in Q pertains to item i. The predicted rating of user U for item i is then computed by $r'_{U,i} = p_U q_i^{\mathrm{T}}$. Given the known ratings in S, the latent matrices P and Q are obtained by solving the optimization problem:

[3] http://gizmodo.com/5991070/big-data-brokers-they-know-everything-about-you-and-sell-it-to-the-highest-bidder

$$\underset{P,Q}{\operatorname{argmin}} J_S(P,Q)$$

where the loss function $J_S(P,Q)$ with regularization parameter γ is given by:

$$\boldsymbol{J_S(P,Q) = \sum_{r_{U,i} \in S} \left[\left(r_{U,i} - p_U q_i^{\intercal} \right)^2 + \gamma (\|p_U\|^2 + \|q_i\|^2) \right]}$$

We consider the Stochastic Gradient Descent (SGD) technique for empirical risk minimization, with a learning rate $\boldsymbol{\lambda}$. In this process, $\boldsymbol{P}$ and $\boldsymbol{Q}$ are modified for each $\boldsymbol{r_{U,i} \in S}$ using the following update rules, which are applied iteratively until both $\boldsymbol{P}$ and $\boldsymbol{Q}$ converge:

$$p_U := p_U + 2\lambda \left[\left(r_{U,i} - p_U q_i^{\intercal} \right) q_i - \gamma p_U \right]$$
$$q_i := q_i + 2\lambda \left[\left(r_{U,i} - p_U q_i^{\intercal} \right) p_U - \gamma q_i \right]$$

4.1.2 Maintaining Decentralized User Profiles

One way to avoid centralized retention of user profiles by a MF recommender is to off-load them to the user side. Predicting unknown scores for all unrated items of a user $\boldsymbol{U}$ requires the latent user vector $\boldsymbol{p_U}$ and the latent item matrix $\boldsymbol{Q}$. In a semi-decentralized setting, $\boldsymbol{U}$ can send the user vector $\boldsymbol{p_U}$ to the system, which holds the item matrix $\boldsymbol{Q}$. The system then computes predicted ratings $\boldsymbol{r'_{U,i}}$ and provides the recommendations to $\boldsymbol{U}$. Once the interaction with $\boldsymbol{U}$ is concluded, the system discards $\boldsymbol{p_U}$, as it is not needed for recommendations for other users. This eliminates the need for a permanent centralized retention of all ratings of all users, and mitigates privacy concerns related to misuse of personal data "at rest".[4]

One drawback of this setting is that the user's $\boldsymbol{p_U}$ may not reflect recent ratings provided by other users. In MF, $\boldsymbol{p_U}$ is affected indirectly by ratings of other users, as new ratings gathered by the recommender affect $\boldsymbol{Q}$ and, in turn, affect vectors of users, who previously rated those items. Fixing latent user vectors and keeping them on the user side hinders these updates of $\boldsymbol{p_U}$, and can cripple the accuracy of the system. Folding-in techniques [18] circumvent this drawback, using an up-to-date matrix $\boldsymbol{Q}$ to re-evaluate the user vector $\boldsymbol{p_U}$ by solving $\boldsymbol{\operatorname{argmin}_{p_U} J_{S_U}(p_u, Q)}$. This optimization problem can be solved with the Stochastic Gradient Descent (SGD) technique by fixing $\boldsymbol{Q}$ and only applying the update rule to $\boldsymbol{p_U}$ above.

4.1.3 Maintaining the Item Matrix

In addition to generating recommendations, the system needs to update the latent item matrix Q based on new ratings. In a standard MF setting, the user and item vectors are typically updated simultaneously. But this cannot be done in the semi-decentralized setting, as no user ratings are retained. Alternatively, following the approaches outlined in [19] [18], folding-in can be applied to update the item vectors by fixing the user matrix P, and retraining an item vector q_i through applying SGD and the respective update rule from above to all the available

[4] We assume in this context that the recommender system is trustworthy, and will indeed discard the user profile following the exchange. For example, service providers may be incentivized to act in a trustworthy manner to maintain user trust and their reputation, and to comply with a published privacy policy. There are also technical solutions that may enforce this, such as leveraging Trustworthy Computing Platforms, or cryptographically protected protocols **[29]**, but they are out of the scope of this work.

ratings. While effective, this iterative update of item factors is also inapplicable unless all the ratings for i are available. Instead, we propose a modified process for updating Q, which leverages the fact that for each new rating $r_{U,i}$ the update of q_i requires access only to the vector p_U of the user who provided the rating. First, the user u sends $r_{U,i}$ along with previous ratings in S_U to the system, which reconstructs p_U. Then, q_i's update rule is applied to the newly added rating and, finally, the system keeps the updated latent item vector and discards both p_U and S_U.

We consider two variants of this scheme. In the first variant, denoted by *new*, we perform an update only with the newly added rating $r_{U,i}$. In the second variant, denoted by *rated*, we take advantage of the availability of the entire set of ratings S_U and apply an iterative update rule to the item vectors of all the items that were rated by U. We note that unlike the *new* variant, which updates one latent vector for every new rating, the *rated* variant updates the vectors of all previously rated items, which may bias the latent item matrix Q towards ratings of highly active users, who provided many ratings. However, this bias can be mitigated, e.g., by weighting the update according to the number of ratings that a user provided.

4.1.4 Evaluation

Full evaluation of the proposed approach was conducted in [20]. We show here two main results, based on evaluation with the Netflix 10M dataset. Figure 6 depicts the accuracy achieved by different recommendation schemes for a time-based split of training and test data. The two proposed variants of the semi-decentralized MF, which update the latent item vectors of either all the rated items (*rated*) or of the recently rated items only (*new*), are compared to four baseline approaches:

- Baseline: Standard centralized implementation of MF, using SGD and retaining all the available ratings [10].
- Static: Similar to baseline, but using only the static matrices Q_s and P_s available at a certain time of transition to "no retention" mode, t_s, to generate recommendations. This mimics a "lazy" approach, where the recommender disregards new ratings and does not update P and Q after time t_s.
- Online: Centralized online MF proposed by Rendle and Schmidt-Thieme in [19], which allows an incremental update of Q by iterating over all the stored ratings. We tuned the parameters using an offline evaluation to $d = 10$ factors (the number of columns in P and Q), learning rate $\lambda = 0.01$, and $n_{iter} = 200$ iterations.
- Slope-one: A simplified item-based Collaborative Filtering (CF) that assumes a linear relationship between the items [21]. This algorithm provides a realistic lower bound for acceptable performance of a personalized recommender. This approach also retains all the available user ratings.

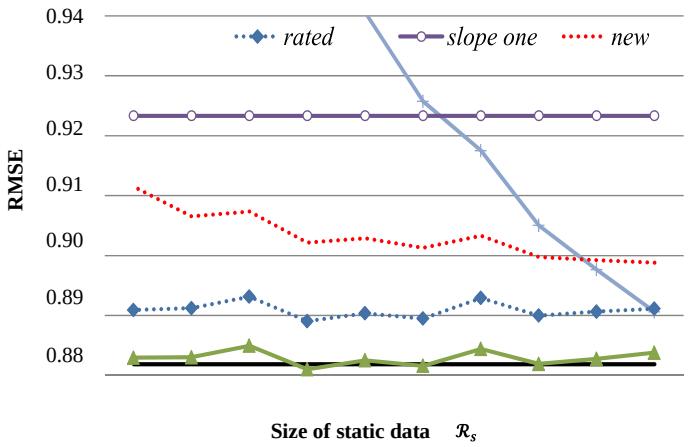


Figure 6 Evaluation with Netflix 10M dataset, time-based split

We denote by R_s the ratio between the number of ratings collected before time t_s, i.e., ratings used to derive Q_s, and the overall number of ratings processed in the evaluation. R_s represents the relative volume of the static initialization data. A special case of $R_s = 0$ reflects a scenario in which the semi-decentralized recommender has no prior user data and starts from the outset with a randomly initialized Q.

As expected, standard centralized MF achieves the highest accuracy. However, MF is not sufficiently scalable and cannot be deployed in a practical Web-based recommender, since it requires the latent vectors to be re-trained for every new rating. In the following analysis, we consider the RMSE of standard MF as the lower bound for error achievable by other approaches and refer to its accuracy as the baseline. *Online* is very close to MF and is consistently superior to our solution. However, it should be noted that online benefits from access to all the available ratings and can re-train the latent factors, which explains its performance. The *rated* variant outperforms *new* across the board: the repetitive updates of the latent vectors of all the rated items in S_U keep Q closer to Q^*(the solution obtained by centralized MF) as rating of users are processed over time. In contrast, the more conservative update process of *new* performs worse when ratings are fed in temporal order, since it is slower to incorporate ratings of new items, which are more important to accurately predict ratings in this setup. We conclude that, when taking a realistically behaving temporal factor of ratings into consideration, the *rated* variant is the best performing semi-decentralized MF approach, and its performance is stable across various values of R_s. Placing *rated* on the range between the upper bound of accuracy set by the baseline MF and the lower bound set by *slope one*, we highlight that *rated* is much closer to MF than to *slope one*. We conclude that *rated* offers a reasonable compromise, as it performs online updates of the latent vectors, without retaining user data.

We also investigated the performance of the proposed approach in a realistic large-scale time-aware scenario. We performed this experiment using the complete Netflix 100M ratings dataset, to mimic the evolution of a Web-scale recommender system that transitions to the semi-decentralized MF model. We compared the accuracy of the two semi-decentralized

variants that do not retain user data with the baseline MF. In this experiment, we sorted all the available Netflix ratings according to their timestamps. We then used the increasing-time window evaluation methodology with a window size of 1 month [22]. That is, we used the data of the first $n - 1$ months as the training set, predicted the ratings of the n-th month, and computed the RMSE for that month. Then, we added the data of the n-th month to the training set and predict the ratings of the $n + 1$-th month, and so on. The overall span of Netflix ratings is 72 months and we initialized the training set with the first four months of data.

Figure 7 shows the RMSE of the *new*, *rated*, and *MF* recommenders (values on the left axis), as well as the relative difference between the *rated* variant and the *MF* baseline (right axis). Initially (months 4 to 24), the difference hovers around the 2% mark, and it steadily diminishes later on, as the training window size increases. Eventually, the difference becomes smaller than 1% around month 55 and virtually disappears from month 68 onward. We note some loss of accuracy (spikes in the relative difference) up to months 38-40. These spikes are due to the introduction of a large number of new items into the system paired with a lower number of users in the system. This makes our algorithms more prone to error, as their update process is slower than that of the baseline (the new approach has a higher spike due to an even slower update cycle).

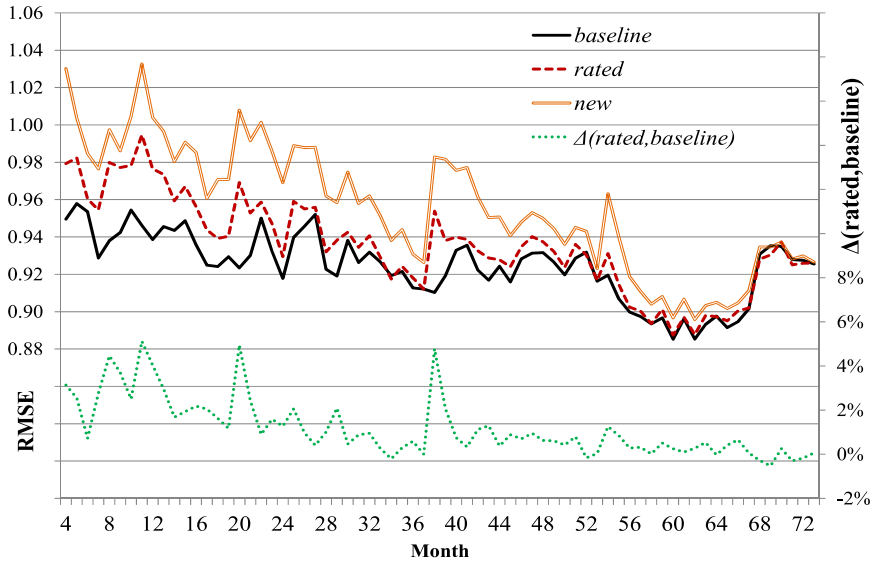


Figure 7 Difference between the rated and new variant and the MF baseline
The bottom line shows the difference between rated and MF

This large-scale simulation clearly demonstrates that the performance of the proposed approaches comes very close to that of the baseline *MF*, as more ratings become available to the recommender. This result is encouraging, as it shows that although avoiding user data retention comes at the cost of accuracy, the cost diminishes over time. Also, while *rated* outperforms *new* at most stages of the simulation, their performance converges at the later stages, as more ratings are included, implying that *new* could also be viable when enough training data is available. Overall, this simulation demonstrates that the proposed semi-decentralized MF variant can be deployed in a practical Web-scale recommender.

4.2 Multi-user privacy-preserving Keyword Search

Another interesting primitive suitable to the UCN environment is privacy-preserving lookup that allows third parties to search for some words in a privacy-preserving manner. Users are assumed to produce highly valuable data for third parties which by exploiting some part of this data can also improve the quality of their service for these users. Therefore, users tend to sacrifice some of their privacy by authorizing third parties to access some selected data (security model 3 – full access) in order to take advantage of the services they use. In this particular case, since the data is outsourced to the cloud (because of some lack of resources at the user side), authorized access is performed through a dedicated privacy-preserving lookup solution.

4.2.1 Background

Several solutions [23, 24, 25] have been proposed to allow the search of words over encrypted data: most of these solutions ensure the privacy of the data and the privacy of the query against the cloud; some of them [23] also ensure that the cloud does not even discover the response to the query. Furthermore, these solutions only allow the owner of the data to perform the lookup request. Very few recent solutions [26, 27] propose the ability to delegate the search operation to authorized third parties; unfortunately, the delegation operation in all of these solutions except the one in [26] is achieved by providing the data encryption key to the authorized parties and hence causing a re-encryption of the entire data in the case of revocation; to the best of our knowledge [26] is one of the first solutions which thanks to the combination of attribute-based encryption and oblivious pseudo random functions, allows a simple and efficient revocation operation of a given third party.

4.2.2 Overview of the proposed solution

As a follow-up to the work in [26] and in order to be in-line with the UCN environment, we propose a preliminary privacy-preserving keyword search primitive that considers a scenario whereby a very large number of users defined as *writers* outsource their encrypted data to the cloud and generate some tokens to the delegated third parties. These third parties which are defined as *readers* perform search operations on data they are authorized to query only. The extension of the original one-to-many architecture into the many-to-many one raises new privacy challenges especially with respect to access patterns. Since a search query (for the same word) targets several different files, an adversary is able to determine the degree of similarity between these different files. In order to ensure access pattern privacy (i.e. ensure that the result of the queries remains confidential), we propose a solution we call MUKS (Multi-User Keyword search) which relies on a third party *proxy* to translate of the query generated by the *reader* into several queries, each of them targeting a different writer's data. Thanks to this *proxy, readers* do not need to receive an authorization per *writer* and they do not need to generate one query per document owned by different *writers*. Furthermore, the adversary model of MUKS naturally considers the proxy as a potential adversary. We therefore analyse the privacy of the solution (data and query privacy) against both the cloud and the proxy while assuming that these two parties do not collude.

We now provide a sketch of the preliminary version of MUKS; the solution will be described in detail in D4.3. Similarly to [26] and [23], the proposed solution uses Private Information

Retrieval as a building block for the search operation. Given the authorizations generated by the *writers*, the *proxy* transforms a PIR query into several PIR queries, one per *writer*, and sends these to the cloud. This *Query.transform* operation is achieved using Elliptic Curve Cryptography and bilinear pairings. The cloud processes this PIR query, generates a response for each of the transformed queries, and encrypts them with the *reader's* public key. The proxy further forwards the responses to the *reader*. Thanks to the use of PIR, neither the cloud nor the proxy can discover the content of the queries. Moreover, because of the encryption of the response by the cloud, the proxy (in addition to the cloud) can no longer infer any information on the access patterns.

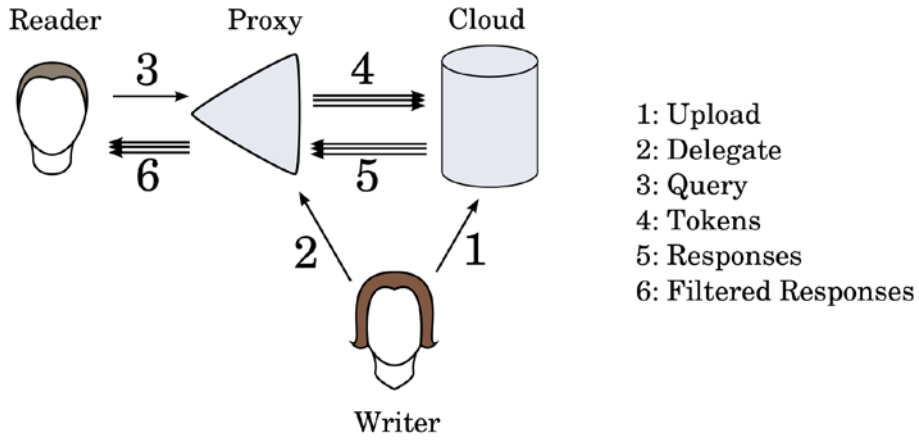


Figure 8 Overview of the Multi-User Keyword Search solution

4.2.3 Integration within UCN

As explained in the previous section, the solution defines four parties: the end-user who will in fact correspond to the *PIH*, the *Cloud* where the data collected at the PIH is outsourced once encrypted, the *third parties* such as a recommender, which will query this data and finally the *Proxy* which will transform the query of the *third party* into several individual queries, such that each query is targeting a specific user's (i.e. writer's) data. The *PIH*'s main task is to encrypt users' data according to their privacy preferences and to generate some tokens for each *third* party authorized to query the corresponding data. The *PIH* may also contact the *Cloud* and the *Proxy*, whenever the end-user wishes to revoke a *third party*. Authorized *third parties* can generate some lookup query for a given word and a set of users. The lookup query is forwarded to the *Proxy* that transforms it into a number of individual queries and transmits these to the cloud. Thanks to the proposed privacy preserving lookup solution, the *Cloud* processes the encrypted data, creates a response to each proxy query and sends these back to the *Proxy* that further forwards them to the *third party*.

4.3 Sticky Policy for Access Control in UCN

The generated information in an Internet of Things (IoT) scenario may have very strict security requirements and so, may need an end-to-end encryption method. The user (data producer) may want to share his data with other entities and specify explicitly who has access to it. Encrypting the information with different keys for different ends may hinder the overall

performance, mainly in an IoT scenario where a user may want to share his data with hundreds of entities.
A mechanism for sending/sharing information, using the concept of sticky policies, will allow the user to store the information in an untrusted environment. Through the use of cryptographic techniques, a link is established between the encrypted information and their access control policies. The system ensures that any third party trying to access the information has to fulfil the set of rules defined by the owner of the information.
We will provide a scalable and granular information access control mechanism, allowing:

- End-to-end information encryption
- Information access to specific elements within a group
- Information secure storage

4.3.1 Description

Sticky policies is a policy based encryption scheme that allows an entity to encrypt data according to a policy file, in such a way that only who meets the policy file requirements has access to the information. It also enables an entity to store data in an untrusted database in a secure way. For example, a system passes a health-care record from a hospital to a research institution and the to a research team. The information might be in a form in which certain attributes such as medical results are encrypted, with an associated sticky policy describing how parts of this could be used - a patient wants his information to be released only to his doctors and requests it be deleted after three years.

Encrypting the data and linking the policies to the key or to the encrypted data enables the distribution of the information to unknown entities while ensuring that only those complying with the associated policies are able to access the information.

The deployment of such a system is reasonably straightforward, as it does not require any change to either existing trusted third parties except to deal with additional policy condition checks or existing storage providers. If the storage providers are used to store users' data, an authenticated reference is passed around instead of the data (see [28])

The policies are defined in XACML 3.0 which allows the expression of very complex rules and various types of Access Control such as: Identity Based Access Control (IBAC), Attribute Based Access Control (ABAC) or Role Based Access Control (RBAC).

Proposed implementation to support ABAC only includes simple rules that allow a user to define what attributes a third party must possess to access the data. For example, a user can define that only the medical staff has access to his blood pressure.

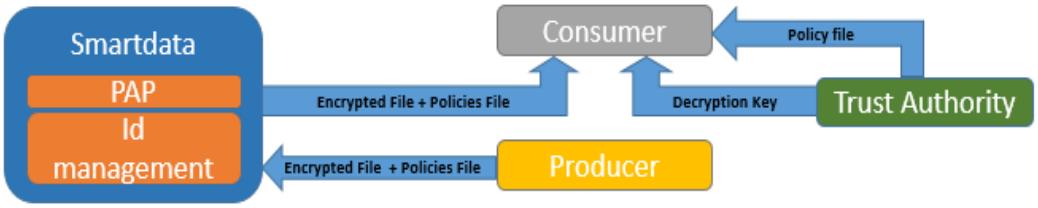
As policies written in XACML are very long, there have been attempts to implement XACLM using JSON. Such an implementation allows maintaining the properties of XACML while reducing the size of policy files.

As stated in [28], *Trusted authorities (*TAs) provide assurance by keeping track of promises the involved parties make to access data, along with controlling access to such data. The TAs' role may be integrated with other functionality, such as being a consumer organization, a

certification authority (CA), or a well-known organization. The TA's role also can be performed by a client-side software component or service that is under the control of users or other parties, or it can be achieved using distributed components or a peer-to-peer mechanism.

4.3.2 Implementation

In the ongoing implementation, it is possible to identify the following entities and flows:



1. Producer: The entity that produces the data to be encrypted;
2. Consumer: Obtains the data and needs to decrypt it;
3. Sticky Policies Service: Service responsible for data encryption/decryption. This service may be:
    a. Internal: If the producer is a non- constrained device, he may be able to encrypt/decrypt the information himself;
    b. External: In the case of constrained devices that cannot encrypt/decrypt data, this service can be deployed as an external service in a more powerful device;
4. Trust Authority (TA): This entity has three main functionalities:
    a. Generate public parameters that are used by the producer to generate keys from policy files;
    b. It serves as a Policy Decision Point (PDP) to verify consumers' authorization;
    c. Generates decryption keys from policy files.
5. Policy Administration Point (PAP): Creates policies with a limited types of rules (still in development phase and slightly immature)
6. Id management: Entity used for authentication purposes; TA uses this entity to obtain consumers' information in order to evaluate authorization.

The consumer and producer entities are generic entities that correspond in UCN respectively to third parties such as recommenders, and to the PIH that acts in the name of an end-user.

4.3.3 Message Encryption

Proposed message encryption flows are presented and described below as well as the associated sequence diagram.

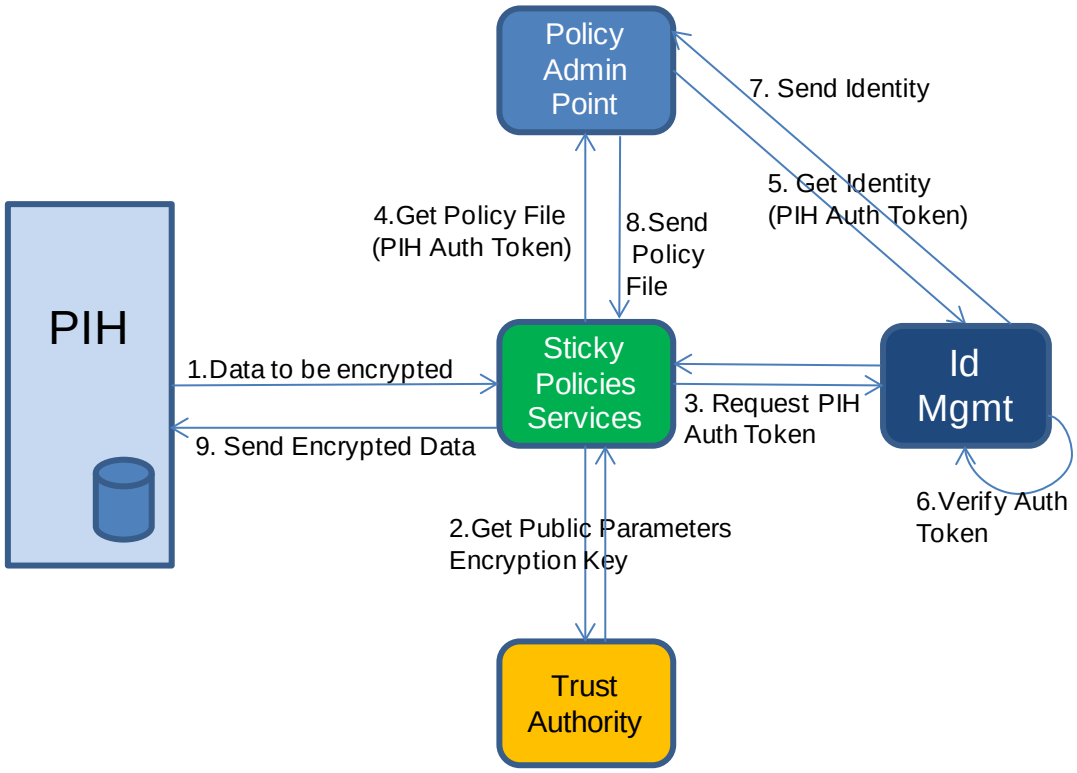


Figure 9 Message Encryption flows with sticky policies

1. The PIH sends the unencrypted data to the Sticky Policies Service;
2. The Sticky Policies Service gets the Public Parameters from the TA;
3. The Sticky Policies Service requests the Auth Token from the Id Management;
4. The Sticky Policies Service sends the PIH Auth Token to the PAP;
5. The PAP sends the PIH Auth Token to the Id Management;
6. The Id Management verifies the token validity;
7. The Id Management sends the PIH identity to the PAP;
8. The PAP sends the policy file to the Sticky Policies Service;
9. The Sticky Policies Service sends the encrypted data to the PIH:
    a. Generates a symmetric cipher;
    b. Encrypts the data with the symmetric key;
    c. Encrypts the symmetric key with the sticky policies key;

Implementation Notes:

- Proposed implementation uses AES as a symmetric cipher, this is one of the most popular symmetric ciphers among security providers.
- The sticky policies key is obtained using the public parameters and the policy file using pairing based cryptography (Java library – jPBC.)
- The Auth token is created using Oauth v2.0
- The sticky policies service is a Java library that can be (preferably) installed in the producer's device or, if it is a computational constrained device, available as a private network service.

- The entities requiring policy files and keys for encryption should register in the Id Management.
- All API requests must have an Auth token.

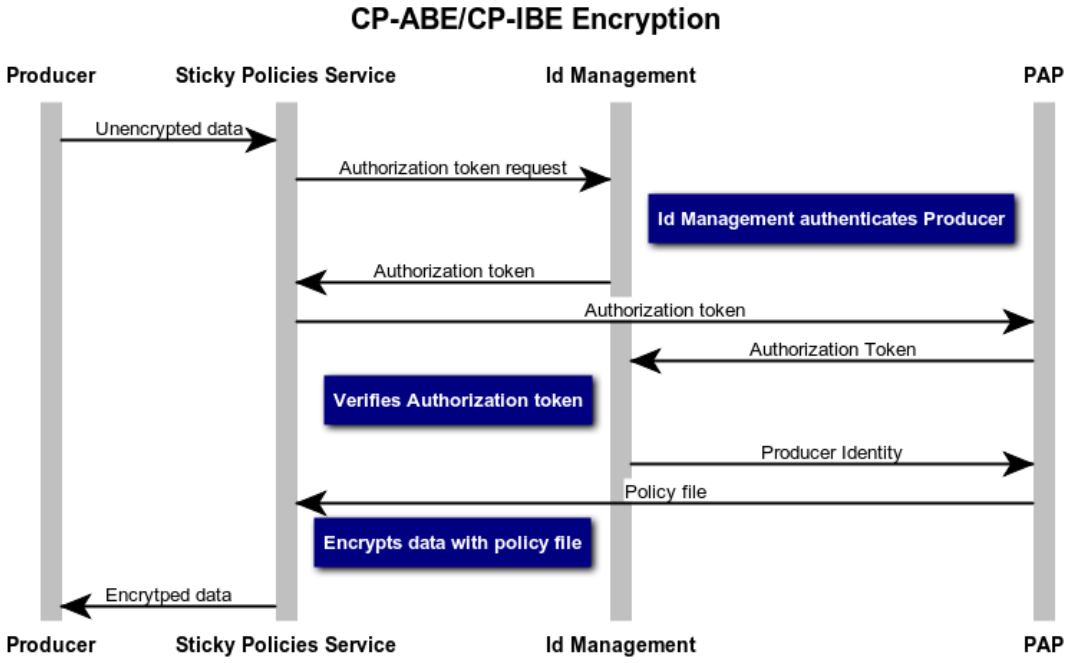


Figure 10 Sequence diagram of sticky policy encryption

4.3.4 Message Decryption

Proposed message decryption flows are presented and described below as well as the associated sequence diagram:

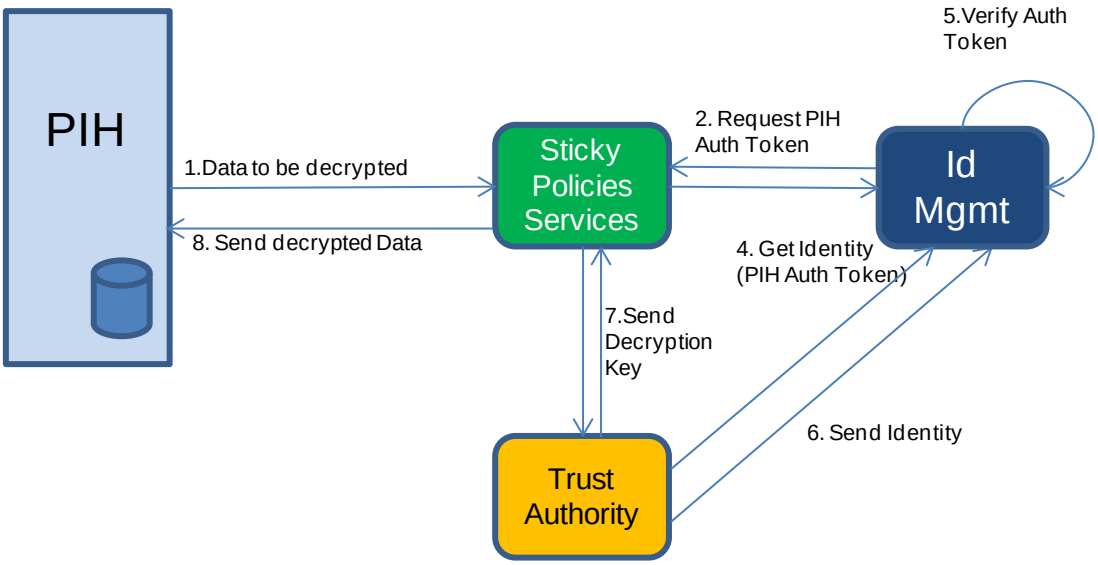


Figure 11 Message decryption flow with sticky policies

1. The PIH sends the encrypted data to the Sticky Policies Service;
2. The Sticky Policies Service obtains the PIH Authorization token from the Id Management;
3. The Sticky Policies Service sends the Auth token and the policy file to the Trusted Authority (TA);
4. The TA sends the Auth Token to the Id Management;
5. The Id Management verifies the Auth Token Validity;
6. The Id Management sends the PIH's identity to the TA;
7. The TA sends the Decryption Key to the Sticky Policies Service;
    a. The TA will verify if the consumer meets the requirements specified in the policy file;
    b. The TA generates a decryption key based on the policy file received;
8. The Sticky Policies Service sends the decrypted data to the PIH:
    a. Extracts the symmetric key from the received data;
    b. Decrypts the symmetric key using the key sent from the TA;
    c. Decrypts the data using the decrypted symmetric key;

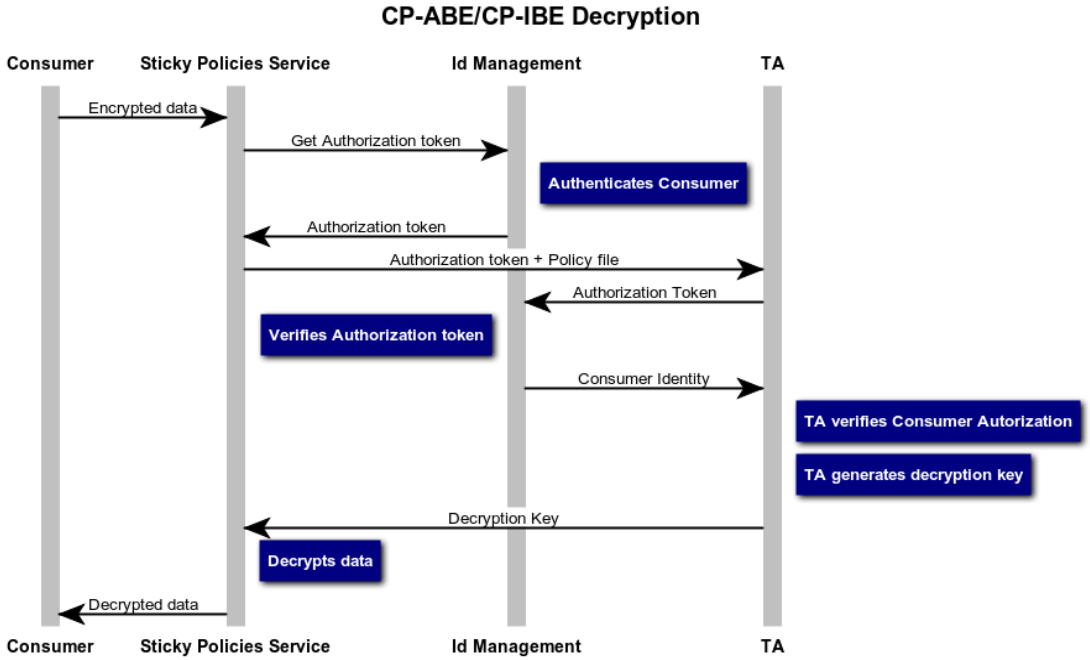


Figure 12 Sequence diagram of Sticky policy decryption

Implementation Notes:

During the decryption process, the TA also acts as a Policy Decision Point (PDP). Java library Balana is responsible for making the decisions based on the policy file received and the attributes of the consumers.

5 CONCLUSION

This deliverable introduced a set of novel privacy preserving primitives for data collection and data processing dedicated to the partial and the full access models defined in D4.1 [2]. Within the partial access model, the proposed solutions use either differential privacy to obfuscate end-users' data without sacrificing the accuracy of the recommender, or secure multi-party computation to design privacy preserving data aggregation that enables third parties to derive some statistics over a large set of users' data without disclosing users' individual inputs. The full access model corresponds to the scenario where the end-user allows authorized parties to have access to his data for a certain period of time. The first solution under this model leverages some matrix factorization technique to design a semi-distributed recommendation algorithm that does not retain users' data in a centralized storage. The secondly proposed mechanism assumes that users' data is outsourced to a storage server after its encryption due to the lack of resources at the PIH, and that users authorize this storage server to operate over their encrypted data for keyword search. The last solution uses sticky policies so that authorized third parties (recommenders) can retrieve data from the PIH based on the rules defined by end-users in the PIH. The deliverable only presents a preliminary version of the two last techniques and their full complete design and integration to the UCN architecture will be described in the next deliverable.

6 REFERENCES

[1] "Specification of the User Context Metrics," UCN deliverable D2.1, 2014.

[2] "Requirements, Ethics and Security Models for privacy preserving data management," UCN Deliverable D4.1, 2014.

[3] "Preliminary UCN Use cases and Applications," UCN deliverable D5.1, 2014.

[4] A. Shikfa, M. Önen and R. Molva, "Broker based private matching," in *PETS*, Waterloo, CA, USA, 2011.

[5] E. Shi, T.-H. Hubert Chan, E. G. Rieffel, R. Chow and S. D., "Privacy-Preserving Aggregation of Time-Series Data," in *NDSS*, 2011.

[6] M. Joye and B. Libert, "A scalable scheme for privacy-preserving aggregation of time-series data," in *Financial Cryptography*, 2013.

[7] I. Leontiadis, K. Elkhiyaoui and R. Molva, "Private and dynamic time-series data aggregation with trust relaxation," in *CANS*, Heraklion, Crete, Greece, 2014.

[8] R. Agrawal and S. Ramakrishnan , "Privacy-Preserving Data Mining," in *SIGMOD Conference*, 2000.

[9] D. Agrawal and C. C. Aggarwal, "On the Design and Quantification of Privacy Preserving Data," in *PODS*, 2001.

[10] Y. Koren, R. Bell and C. Volinsky, "Matrix Factorization Techniques For Recommender Systems," in *IEEE Computer Society*, 2009.

[11] C. Dwork, F. Mcsherry, K. Nissim and A. Smith, "Calibrating Noise to Sensitivity in Private Data Analysis," in *Proc. 3rd Theory of Cryptography Conference*, 2006.

[12] T. Kandappu, A. Friedman, R. Boreli and V. Sivaraman, "PrivacyCanary: Privacy-Aware Recommenders with Adaptive Input Obfuscation," in *MASCOTS*, Paris, France, 2014.

[13] A. Narayanan and V. Shmatikov, "Robust De-anonymization of Large Sparse Datasets," in *IEEE Symposium on Security and Privacy*, 2008.

[14] U. Weinsberg, S. Bhagat, S. Ioannidis and N. Taft, "BlurMe: inferring and obfuscating user gender based on ratings," in *RecSys*, 2012.

[15] A. Chaabane, G. Acs and M. Kaafar, "You Are What You Like! Information Leakage Through Users' Interests," in *NDSS*, 2012.

[16] M. Kosinski, D. Stillwell and T. Graepel, "Private traits and attributes are predictable from digital records of human behavior," *Proceedings of the National Academy of Sciences*, pp. 1091-6490, March 2013.

[17] G. Greenwald and E. MacAskill, "NSA Prism program taps in to user data of Apple, Google and others," *The Guardian*, 7 June 2013.

[18] B. Sarwar, G. Karypis, J. Konstan and J. Riedl, "Incremental Singular Value Decomposition Algorithms for Highly Scalable Recommender Systems," in *Fifth International Conference on Computer and Information Science*, 2002.

[19] S. Rendle and L. Schmidt-Thieme, "Online-updating regularized kernel matrix factorization models for large-scale recommender systems," in *RecSys*, 2008.

[20] D. Vallet, A. Friedman and S. Berkovsky, "Matrix Factorization without User Data Retention," in *PAKDD*, 2014.

[21] D. Lemire and A. Maclachlan, "Slope one predictors for online rating-based collaborative filtering," *Society for Industrial Mathematics,* vol. 5, pp. 471-480, 2005.

[22] P. Campos, F. Díez and I. Cantador, "Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols," *UMUAI*, 2013.

[23] E.-O. Blass, R. Di Pierto, R. Molva and M. Önen, "PRISM- Privacy-Preserving Search in MapReduce," in *PETS*, 2012.

[24] Y.-C. Chang and M. Mitzenmacher, "Privacy preserving keyword searches on rome encrypted data," in *Proceedings of Applied Cryptography and Network Security (ANCS)*, 2005.

[25] D. Boneh, G. Di Crescenzo, l. P. Ostrovsky and G. Persiano, "Public key encryption with keyword search," in *EUROCRYPT*, 2004.

[26] K. Elkhiyaoui, M. Önen and R. Molva, "Privacy preserving delegated word search in the cloud," in *SECRYPT*, 2014.

[27] R. Curtmola, J. Garay, S. Kamara and R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions," in *ACM Conference on Computer and Communications Security (CCS)*, 2006.

[28] S. Pearson and M. C. Mont, "Sticky policies: An approach for Managing Privacy across Multiple Parties," *Computer,* vol. 44, no. 9, pp. 60-68, 2011.

[29] V. Nikolaenko, S. Ioannidis, U. Weinsberg, M. Joye, N. Taft and D. Boneh, "Privacy-preserving Matrix Factorization," in *CCS*, 2013.

[30] G. P. and W. Wang, "Using Large Scale Distributed Computing to Unveil Advanced Persistent Threats," New York, NY: AT&T Security Research Center, 2012.

[31] Y.-F. Lu, C.-F. Kuo and A.-C. Pang, "A half-key key management scheme for wireless sensor networks," in *ACM Symp. on Research in Applied Computation,RACS*, 2011.

[32] P. Giura and W. Wang, "Using Large Scale Distributed Computing to Unveil Advanced Persistent Threats," New York, NY: AT&T Security Research Center, 2012.