

C3P: Context-Aware Crowdsourced Cloud Privacy

Hamza Harkous, Rameez Rahman, and Karl Aberer

École Polytechnique Fédérale de Lausanne (EPFL)
hamza.harkous@epfl.ch, rrameez@gmail.com, karl.aberer@epfl.ch

Abstract. Due to the abundance of attractive services available on the cloud, people are placing an increasing amount of their data online on different cloud platforms. However, given the recent large-scale attacks on users data, privacy has become an important issue. Ordinary users cannot be expected to manually specify which of their data is sensitive, or to take appropriate measures to protect such data. Furthermore, usually most people are not aware of the privacy risk that different shared data items can pose. In this paper, we present a novel conceptual framework in which privacy risk is automatically calculated using the sharing context of data items. To overcome ignorance of privacy risk on the part of most users, we use a crowdsourcing based approach. We use *Item Response Theory* (IRT) on top of this crowdsourced data to determine the sensitivity of items and diverse attitudes of users towards privacy. First, we determine the feasibility of IRT for the cloud scenario by asking workers feedback on *Amazon mTurk* on various sharing scenarios. We obtain a good fit of the responses with the theory, and thus show that IRT, a well-known psychometric model for educational purposes, can be applied to the cloud scenario. Then, we present a lightweight mechanism such that users can crowdsource their sharing contexts with the server and determine the risk of sharing particular data item(s) privately. Finally, we use the Enron dataset to simulate our conceptual framework and also provide experimental results using synthetic data. We show that our scheme converges quickly and provides accurate privacy risk scores under varying conditions.

1 Introduction

1.1 Motivation and Challenges

Cloud computing platforms have become a ubiquitous presence in our digital lives. Given the pervasiveness of useful cloud services such as storage, online document editing, media streaming, etc., data which would normally be on the user’s local machine, now invariably lies in the cloud. Recent large scale leakage of data [1] has raised serious concerns about users privacy. Prior to designing privacy mechanisms, it is important to identify the challenges of privacy provision in the cloud, which can inform potential solutions. In particular, we notice three major stumbling blocks towards privacy provision in the cloud:

a) Privacy vs Services Dilemma: To tackle privacy concerns, some cloud computing companies provide the users with the option of client-side encryption to protect the data before it leaves the users’ device, thus preventing any

other entity from data decryption, including the cloud provider itself. However, while this approach eliminates most of the data privacy concerns, its main disadvantage is that the user cannot readily utilize existing cloud services. For some services, attempts exist at designing alternatives that operate over encrypted data, benefiting from the recent breakthroughs in homomorphic encryption [2]. In addition to resulting in services orders of magnitude less efficient than their counterparts, homomorphic encryption is provably not sufficient for constructing several essential services involving multiple users [3]. Furthermore, resorting to homomorphic encryption as the ultimate solution requires rewriting most of the cloud applications' code to operate over the encrypted data. New versions of existing L^AT_EXcompilers, photo filters, music recommenders, etc., based on homomorphic encryption, will need to be programmed with the goal of keeping all data private, which is evidently non-realistic.

b) Difficulty of manually assessing data privacy levels: Users cannot be expected to individually assess the sensitivity level for each item before they share it as that can require a lot of investment in terms of time and effort, coupled with technical expertise. A recent survey [4] has shown that, in one out of four organizations, the management has little or no understanding of what constitutes sensitive data. Evidently, this fraction is expected to be significantly higher for individual users.

c) General lack of awareness about privacy: This includes limited notions about privacy being restricted to hiding 'sensitive' content, such as personal identification numbers, credit card details etc. Often, the metadata associated with the data item, the location and device from which the item is shared, the entity with whom the data is shared, etc., can be as important as the content of the data itself.

In our solution for privacy provision in the cloud, we seek to overcome these above hurdles.

1.2 Approach and Contributions

How do we address the 'stumbling blocks' that we identified in Section 1.1? First, we show how we can use a centralized solution to facilitate crowdsourcing for privacy *without requiring revelation of users preferences*. We argue that to achieve this, cryptographic methods are infeasible, and we present a novel design that allows users to reveal their preferences to the central server privately. We show how an existing psychologically grounded method for analyzing users preferences and data properties, can be rigorously used to analyze this crowdsourced information. Users can then reap the benefits of this crowdsourced information as the server analyzes it to provide them with sensitivity indicators when they share new data.

By crowdsourcing the solution, users are no longer isolated individuals who lack privacy awareness. They can now be guided by the *Wisdom of the Crowd*. Also, they do not have to exert manual effort to find the sensitivity associated with each item they share, as the server can guide them automatically. Furthermore, they need not worry about getting stuck with 'bad' crowdsourced information, i.e., about majority of users being as clueless about privacy as them.

This is because the psychometric method we use for analyzing this information, Item Response Theory, ensures that computed parameters of data items do not only apply to a specific sample of people. The solution would ensure, for example, that sharing compromising photos of oneself with the public is deemed risky even when majority of the people in the system are doing so. Only a few conservative users in the system are enough to keep the system risk-averse. Finally, we validate our design with both simulation and empirical data, thus showing the feasibility of our solution.

Specifically, we make the following main contributions in this paper:

- We propose a privacy framework, which is specific to the cloud scenario and incorporates the nuances of data sharing within cloud, such as the *Privacy vs Services Dilemma* and *Lack of Privacy Awareness and Effort* on part of most users.
- We create a realistic vocabulary for a personal cloud, and use it to create ‘Human Intelligence Tasks’ on the *Amazon Mechanical Turk*. We measure people’s responses, in terms of their privacy attitudes, against the *Item Response Theory* (IRT) and find a good fit. We thereby demonstrate that Item Response Theory, a well-used psychometric model for diverse purposes, can be applied fruitfully in the cloud scenario.
- Our solution depends on crowdsourcing the contexts and policies associated with shared items. The sensitivity associated with different items is determined by grouping together same (or similar) contexts and analyzing different policies set by people with different privacy attitudes. However, we also have to ensure the privacy of this aggregated context information. Towards that aim, we provide a lightweight mechanism based on *K-Anonymity* [5] for privately calculating similarity between items in a centralized way, without depending on infeasible cryptographic methods.
- We perform a set of experiments using synthetic data, with various graphs for user activities, item distribution, and types of users (honest vs. malicious).
- Finally, we use the *Enron* email dataset for evaluating our framework. This dataset gives us a good model of users sharing activities and the diversity of data items (and their contexts). Under both datasets, we show that our scheme bootstraps quickly and provides accurate privacy scores in varying conditions.

2 System Model

2.1 Interacting Entities

We consider a system involving interactions between two types of entities: *end-users* and *cloud service providers (CSPs)*. The end-user can play one of two roles: *data sharer* or *data observer* while the cloud provider can only be a data observer. A data sharer is an end-user who shares *data items* she possesses. A data observer is any entity that is given access to observe the shared items by the data sharer.

We assume that the user sends her data to a single CSP, called the *intermediary* that acts as the repository for this user’s data. The user can select to give

other CSPs access to her data through that CSP (e.g. when the latter has an API that the other CSPs can use). The interaction between these two types of entities is in the form of data sharing operations. Each such operation is initiated by an end-user s_0 who shares a data item d (e.g. document, picture, etc.) with a CSP or another user s_1 . Additionally, the data sharer intends from the sharing operation to obtain a certain service of interest from the CSP, such as music streaming, document viewing, file syncing, etc. The network is dynamic, in the sense that these entities can enter and leave the network, and the user items can be shared over time, not necessarily concurrently.

2.2 Threat Model

We assume that the user is interested in hiding her sensitive data from the CSPs. Existing privacy threat models consider an adversary who attempts at discovering quantifiable sensitive information, such as location, browsing history, credit card information, etc. In our model, we do not set an a priori definition of sensitive information due to the heterogeneity of the shared data items we consider. Instead, we develop a protocol that quantifies the sensitivity of a certain sharing operation (determined by its context), based on the privacy policies that people use. Furthermore, we assume that the CSP is *honest but curious*, in the sense it follows the protocol, but it can arbitrarily analyze the protocol transcript offline to infer extra information.

2.3 Our Conceptual Framework

We now discuss the key concepts and components that underlie our conceptual framework for privacy provision in the cloud.

Context Vocabulary In Section 3, we use the notion of *Context vocabulary* to define the contexts of items shared in a given domain. A context accounts for the content features of the items, the metadata associated with it, and the environment of the sharing operation (e.g. data observers, device used, etc.).

Sharing Policy People can share different data items with different policies, where a policy is in the range $[0, 1]$ and 0 signifies full transparency while 1 signifies full obscurity. We discuss this in more detail in Section 3.2.

Crowd-Sourcing In our framework, after each sharing operation, the context of the item and the policy applied are eventually aggregated at the cloud via a privacy preserving mechanism. This aggregation is required so that the *Lack of Privacy Awareness* may be overcome, and individual decisions could be guided by the *Wisdom of the Crowd*.

Risk Evaluation Based on the processing and analysis of the crowdsourced information, the system can guide others about the privacy risk that is posed by sharing different items in different contexts. Towards that aim, we use *Item Response Theory* (IRT) which is a well-known psychometric function that has been widely used in psychology, education, public health, and computer adaptive testing.

Policy Recommendation The final component in our framework is a suite of risk mitigation applications. By this, we mean system recommended policies that can guide the general user in minimizing risk while still availing services.

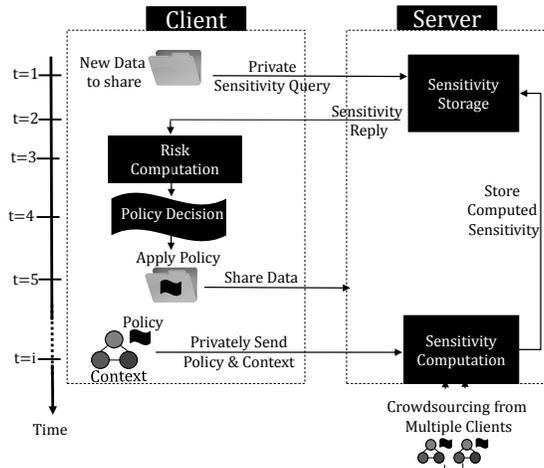


Fig. 1: Sequence diagram of the system

In this work, we do not focus on Policy Recommendation and leave it for future work.

In Figure 1, we show a sequence diagram summarizing the steps taken in one run of the system. The client contacts the server with a private query about the sensitivity of the current context ($t=1$), in a way that the server remains oblivious about the actual context. Upon receiving the response with the sensitivity ($t=2$), the client locally computes the privacy risk of sharing the data ($t=3$) and decides on the relevant privacy policy ($t=4$). Next, the client sends the data at $t=5$. At a later round ($t=i$), the client sends the context along with the used policy after it makes sure that the server cannot associate the context with the actual sharing operation. The server determines the similarity of this item with other items that users have crowdsourced to it. Using psychometric functions, the server computes the sensitivity associated with the item being shared, which is used to respond to future sensitivity queries.

3 Context Vocabulary and Sharing Policies

We begin by describing the fundamental building blocks of our framework, which refer to the context in which an item is shared and the policy with which the item is shared.

3.1 Context Vocabulary

We introduce the technical notion of ‘Context’, which includes the metadata associated with a particular data item, user supplied information about the data item (such as tags), and the environment features in which the data is being shared (such as the device information or the relationship with the observer). Furthermore, ‘Context’ also includes information extracted through content analysis of the data item, such as topic modeling statistics in case of a text document and face recognition in the case of images.

For an illustration of ‘Context’, consider a case where Bob shares a *word document* about *financial risk* authored by *sharer* (i.e. Bob himself) on Bob’s

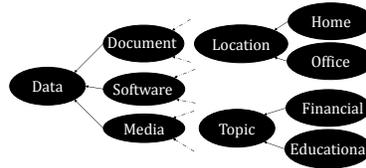


Fig. 2: An example vocabulary for data sharing in the personal cloud

laptop and shared with a *colleague*. The words in italics capture the context of the data item. For a specific domain, ‘Context Vocabulary’ is the set of all fields that can be used to represent any shared item in that domain. Put another way, the context vocabulary is the vocabulary that can be used to represent all possible contexts in a given domain. We give an example of such a vocabulary in Figure 2.

The general template for a context of an item would be a tuple of the general form $(field1=value1, field2=value2, \dots)$, containing f fields. Thus, the context of the data item in the above example would be $(data_type=word\ document, topic=financial\ risk, device=laptop, author=sender, observer=colleague)$.

It should be noted that there are usually two kinds of fields associated with a data item. The first are those which are by default associated with the data item, e.g., *data type*, and other metadata information, e.g., *author*, which are available (or can be extracted by anyone) if the data item is shared completely transparently as in plaintext. We term these *explicit* fields. The second are defined by the sharer while sharing the data item, e.g., *observer*, *topic*, *device*, or other tags that might be associated with the data item. We term these *implicit* fields.

We note here that it is not necessary (or even usual) for all data items to have all context fields available. An item’s context is defined by whatever fields are available. For example if we have a *pdf* file which does not have its *author* present, then obviously the file’s context would not include the author. Put another way, the value of the author field would be considered empty.

3.2 Sharing Policies

When a user decides to share a data item, it does so with a policy. This policy ranges from 0 to 1, where 0 signifies full transparency while 1 signifies full obscurity. For example, if the user decides to encrypt a file, then this would be symbolized by a policy value of 1. On the other hand, sharing an unencrypted file while hiding some meta-data fields (such as e.g., *author*, *modified_by* etc) would result in a policy value between 0 and 1.

4 Crowd-Sourcing and Risk Evaluation

As shown in Figure 1, a client can privately query the server about the sensitivity of a specific sharing operation and get a response based on that. In this section, we describe these parts of our framework in more detail. Informally speaking, the privacy guarantee that we achieve throughout is that, at any time, the server has multiple contexts that can be associated with each sharing operation. Accordingly, the context of each operation is never deterministically disclosed to the server.

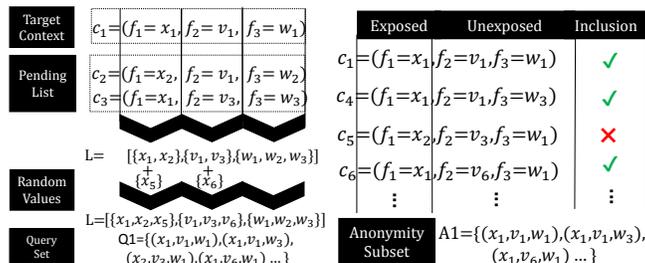


Fig. 3: *QuerySet* formation and anonymity subset definition

4.1 Privacy Aware Querying

Directly sending the context to the server allows it to associate the sharing operation with that context, which we aim to avoid. Instead, we describe a scheme, where the client queries the server about multiple dummy contexts, in a way that hides the actually requested one.

QuerySet Formation We denote by *targetContext* the context for which the client is querying. This context is sent as part of a *QuerySet*, containing other contexts, which we term as *homonyms*. As shown in Figure 3, suppose that the *targetContext* is $c_1 : (f_1 = x_1, f_2 = v_1, f_3 = w_1)$. The client forms a list of alternative values for each field, e.g. $L = [\{x_1, x_2, x_5\}, \{v_1, v_3, v_6\}, \{w_1, w_2, w_3\}]$ so that, in total, each field has k possible values. Then the homonyms are formed by producing the cartesian product of all the sets in L . This results in contexts having different combinations of field values.

The choice of the alternative values is not totally at random. In order to allow *targetContexts* to appear faster in multiple *QuerySets*, thus approaching the privacy condition formalized in this section, the client keeps a *Pending List (PL)*, containing previously queried *targetContexts*. It selects at random a fraction of $t \times k$ values¹ from PL when available and fills the rest of the values from the domain of each field.

The client sends this *QuerySet* to the server. The server, on receiving a *QuerySet*, responds with a subset of all those contexts for which it knows the sensitivity². If the sensitivity of the *targetContext* is also returned by the server, the client decides to apply a policy on the data item based on this; otherwise the client can choose to do the same uninformed. In both cases, the actual data item is sent afterwards to the server. Once the server receives the actual data item, it can easily infer the *exposed* part of the *targetContext*. This part includes those *explicit* fields as defined in Section 3.1, which the client did not choose to hide. It is evident to notice that, by the construction of the *QuerySet*, the server is not able to deterministically infer any field of the *unexposed* part of the context (containing all *implicit* fields and those *explicit* fields which have been hidden by the client). In particular, the server has k possible values for each such field. Moreover, assuming there are u fields in the unexposed part, we will have k^u

¹ k is a constant ($0 < k < 1$) (we take $t = 2/3$ in our experiments).

² We shall discuss how the server calculates this sensitivity in Section 4.2.

contexts that match the exposed part of the *targetContext*. We call this set of contexts the *Anonymity Subset* (A_i) of the *targetContext* c_i , and we illustrate its contents with an example in Figure 3. With respect to the server, one of the elements of this subset is the *targetContext*, but no element can be ruled out without further information.

We now add the following definition:

Definition 1. We say that a context c can be **validly associated** with the sharing operation of item d_i if c has appeared in A_i and if the server cannot assert with certainty that c exclusively belongs to one or more anonymity subsets other than A_i .

Hence, at this stage, we have the following guarantee:

Guarantee 1. At the querying phase, the server receives k^u contexts that can be validly associated with the current sharing operation.

Crowdsourcing Up till now, we have shown how the client privately queries the server about the sensitivity. In order to compute this sensitivity, the server relies on crowdsourcing, through *privately* collecting *targetContexts* along with the corresponding policies (together called the *Crowdsourcing Information* (CI)) from different clients. We alternatively say that a context c is *crowdsourced* when $CI(c)$ is sent to the server. The client should not send dummy information as in the querying phase in order to not affect the accuracy of the sensitivity computation. Thus, we now present the scheme in which client sends the CI in a way that continues to maintain Guarantee 1 for all the sharing operations. As a result, the server will be able to know, for example, that a client *Bob* shared a *financial document* with a *colleague* in an *plaintext form*, but it will not be able to link the document topic or his relationship with the observer to a specific sharing operation.

One way this guarantee might be weakened is if the client sends the CI in a way that allows the server to discover the anonymity subset in which the context was the *targetContext*. For example, sending $CI(c)$ after c has appeared in a single anonymity subset A_1 will reveal to the server that c corresponds to data d_1 . Hence, the first intuitive measure for preventing this association is to wait until a context appears in multiple anonymity subsets before sending the CI .

However, this measure is not sufficient. Consider the case of two contexts c_x and c_y , both only appearing in anonymity subsets A_4 and A_6 . Suppose that we require that a context appears in at least two anonymity subsets before it is sent. Then, both $CI(c_x)$ and $CI(c_y)$ will be sent directly after item d_6 (with anonymity subset A_6) is sent. At this point, the server is sure that one of c_x and c_y is the *targetContext* for A_4 and the other for A_6 . All of the other $k^u - 2$ contexts that have appeared in A_4 and A_6 are no more possible candidates for being the actual *targetContext* from the viewpoint of the server. Hence, Guarantee 1 for these two sharing operations is weakened as the $k^u - 2$ contexts are now deterministically associated with other anonymity subsets. The guarantee will be weakened further if there was a third item d_8 that has been subsequently

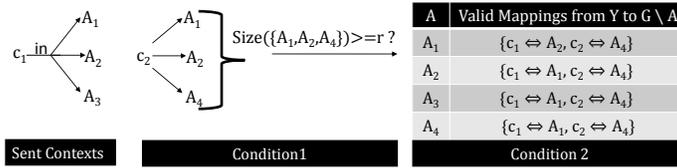


Fig. 4: Checking privacy conditions before crowdsourcing context c_2

sent, with its context c_8 appearing in A_4 and A_8 . From the server's viewpoint, A_4 is no more a valid possibility for c_8 due to the mapping deduced when c_x and c_y were sent. Therefore, the server can deterministically associate A_8 to c_8 , and the whole context for d_8 is revealed. The main weakness in this naive method is that it does not account for the fact the server can link multiple sending instances and reduce the possibility of mapping to a single case. Our strategy to counteract that and keep Guarantee 1 is to verify that crowdsourcing the next context preserves the property that each sent context item is still validly associated with all the anonymity subsets it has appeared in.

At this point we add another definition:

Definition 2. We say that there is a **valid mapping** from a list of contexts to a list of anonymity subsets if each context in the former can be validly associated with a distinct anonymity subset from the latter.

Suppose the client has just completed the sharing operation i , and is attempting to crowdsource the contexts that have not been sent yet, which are kept in its *Pending List* (PL_i). We also denote by SL_i the *Sent List*, containing all contexts that have been crowdsourced previously, and by G_i the group of all client's anonymity subsets up to (and including) A_i . Towards achieving Guarantee 1, a context $c \in PL_i$ can be crowdsourced only when the following two conditions are true:

1. c appears in at least r anonymity subsets
2. For each $A \in G_i$, there exists a valid mapping from the list $SL'_i = SL \cup \{c\}$ of contexts to the list $G_i \setminus A$ of anonymity subsets.

Going back to the previous example, after each of c_x and c_y has appeared in two anonymity subsets, condition 1 is satisfied. However, condition 2 is not satisfied since excluding A_4 will lead to $G \setminus A_4 = \{A_6\}$, and then we cannot map each context to a distinct anonymity set.

Figure 4 illustrates with another example how the two conditions can be verified. For each *targetContext*, the client maintains a list of the anonymity subsets it has appeared in. In addition, it maintains two lists: $U1$, containing the *targetContexts* that have not satisfied the first condition yet³, and $S1U2$, containing the list of items that have satisfied the first condition but not the second. The figure shows a valid mapping that exists for each anonymity subset in G when c_2 is considered for crowdsourcing. It is worth noting that $PL = U1 \cup S1U2$. Also, as discussed in Section 4.1, when the contexts of PL appear in more anonymity subsets, the above privacy conditions will be satisfied faster; hence, they were used in the construction of the *QuerySet*.

³ regardless of whether the second condition is satisfied

Lemma 1. *Checking conditions 1 and 2 allows to preserve Guarantee 1.*

Proof. Consider any context that is about to be crowdsourced. Condition 2 implies that for each $A \in \mathbb{G}$, there is a possibility that the *targetContext* of A has not been sent yet. Hence, each context in $c \in SL'$, can still be *validly associated* with all the r subsets it appeared in. Let Z_i be the list of all contexts that appeared in the elements of \mathbb{G}_i . It is evident that there is no new information being sent about the contexts in $Z \setminus SL'$. Therefore, all the contexts in Z_i can still be validly associated with the anonymity subsets they appeared in. Accordingly, Guarantee 1 is preserved. \square

Discussion: We note that an alternative scheme for crowdsourcing that includes encrypting the context before sharing it would not work. In our framework, the server is required to use a similarity function to match the context with other ones sent by people in order to compute the context sensitivity. Even if we encrypt the context before we send it, the server will be able to know it by computing its similarity with all the possible contexts in the vocabulary (as the latter are not large enough to prevent being iterated over easily). Another place where encryption might be applied is at the querying phase, where *Private Information Retrieval (PIR)* techniques with constant communication complexity might replace the *QuerySet* technique. However, as the complexity gain is absent, and the privacy guarantee obtained by the querying phase is limited by the crowdsourcing phase, we do not resort to the encryption-based method, which is more complex to implement.

4.2 Sensitivity and Risk Evaluation

When the server receives the crowdsourcing information, it seeks to determine the sensitivity associated with this item based on same or similar items shared with different policies in the past by different users. The client, upon receiving this sensitivity, locally computes the privacy risk of sharing. In this paper, for computing the sensitivity, we use *Item Response Theory (IRT)*, a well-known psychometric function, which we describe next.

Sensitivity Computation by the Server Item Response Theory (IRT) is a modern test theory typically used for analyzing questionnaires to relate the examinees' probability of answering a question correctly (or in general a correct response probability P_{ij}) to two elements: (1) the difficulty of the question (or in general a latent threshold parameter β_i of item i) and (2) the examinees' abilities to answer questions (or in general a latent parameter θ_j for each person j). In contrast to *Classical Test Theory (CTT)*, which measures a person's ability based on averages and summations over the items, IRT has two distinguishing features: (1) the group invariance of calculated item parameters (i.e. a single item's parameters do not only apply to the current user sample, assuming the social norms won't vary significantly) and (2) the item invariance of a person's latent trait (i.e. the trait is invariant with respect to the items used to determine it) [6].

In this work, we apply IRT by mapping the item's difficulty to the sensitivity, the user's trait to the privacy attitude (or willingness to expose the items), and

the response probability to the policy level of the item (similar to previous works [7, 8]).

We focus on the unidimensional IRT models, which make three main assumptions about the data: (1) unidimensionality (i.e. there is a single underlying trait θ that determines the person’s response), (2) local independence (i.e. for each underlying trait θ , there is no association between responses to different items), and (3) model fit (i.e. the estimated item and person parameters can be used to reproduce the observed responses) [9]. An IRT model is termed as *dichotomous* if the responses to the questions are binary ones (correct/incorrect) and *polytomous* if there are multiple levels of the response (e.g. a five-level Likert scale with responses: strongly disagree/disagree/neutral/agree/strongly agree).

The Rasch model, one of the most common IRT models, assumes that the probability of correct response is a function of θ and β only and that the items are equally discriminative for testing the underlying trait. It is particularly advantageous with smaller sample sizes, due to its simplicity and few parameters, and, as we show in Section 5.1, it also fits well in the scenario of cloud data sharing. The parameters of the dichotomous Rasch model for an item i and a person with parameter θ are related by the following function, called the *Item Response Function (IRF)*:

$$P_i = \frac{1}{1 + e^{-(\theta - \beta_i)}} \quad (1)$$

With polytomous models, we will make the assumption that the policies chosen by the users are on the same scale for all the items. It is similar to the case of Likert scale, where the same set of categories are applied for each item in the test. Accordingly, the most suitable model for us, and whose fit to the cloud scenario will be demonstrated in Section 5.1, is the Rasch Rating Scale Model. For estimating the parameters of the different models, we used *Marginal Maximum Likelihood estimation*, which is an expectation-maximization algorithm. The estimation technique relies on having enough responses for multiple items by different people. For more details about item response theory models, the reader is referred to the following works [6, 9, 10].

Risk Computation by the Client The sensitivity is an indication of the magnitude of privacy loss incurred when data is lost. The client can combine this measure with another measure of the *likelihood* that this event happens, using information that is kept locally, such as the level of *trust* for the current observer, the level of *protection* (i.e. the policy), etc. The privacy risk is then a combination of the sensitivity and the likelihood.

5 Evaluation and Experiments

5.1 Experiments for Validating IRT

Since we shall be using Item Response Theory (IRT) to calculate the sensitivity of shared items, the first question that needs to be answered is this: *Can IRT be meaningfully applied in the cloud scenario in which people share data items in a variety of contexts?* In order to investigate this and to empirically ground

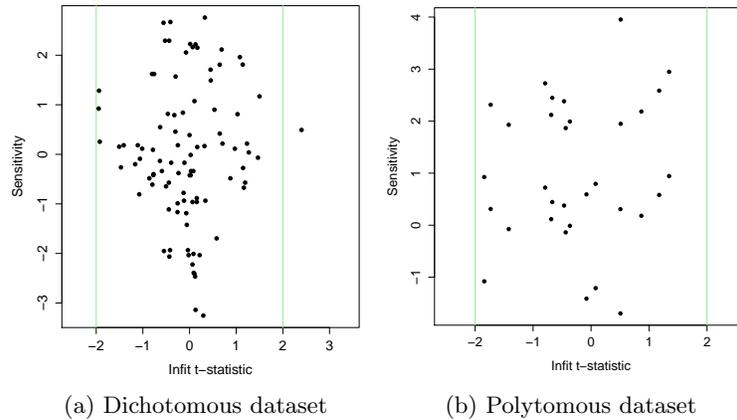


Fig. 5: Bond-and-Fox Pathway Map on the mTurk data (a dot represents a context item)

our design and subsequent experiments, we validated IRT for the cloud scenario using real people’s feedback on Amazon Mechanical Turk. Next, we explain our methodology for this validation.

Methodology We created a realistic vocabulary for the personal cloud, and, based on it, we developed a list of questions that we submitted as *Human Intelligence Tasks* (HITS) on Amazon mTurk⁴. We created two separate HITS for the dichotomous and polytomous cases of IRT. For the dichotomous case, we asked 96 questions to which we received answers from 81 people. For the polytomous case (with 3 categories), we asked 16 questions to which we received answers from 50 people⁵. Here each question represents a *context item* while the users responses represent their *policies* for sharing in the given context.

We analyzed the results using the eRm Package in R [12]. For testing the model fit, we used the standardized (STD) and the mean square (MSQ) infit statistics. An infit statistic is a weighted fit statistic derived from the squared standardized residual between the observed data and the one predicted by the model [13]. The STD infit indicates whether the data fits the model *perfectly* and is also an approximate t-statistic. In Figure 5, we show the STD infit statistic in the two cases of dichotomous and polytomous items, along with the sensitivity value of items (threshold values in the polytomous case) in each graph, also called the *Bond-and-Fox Pathway Map*. We notice that all the values in the polytomous case and all but one in the dichotomous case lie between -2 and 2, which are the typically acceptable bounds [13]. We also derived the MSQ infit which serves as an indication of whether the data fits the model *usefully*, i.e. if it is productive for measurement. We found that the MSQ infit was in the

⁴ The vocabulary and the survey are available online: <http://goo.gl/xjuvuj>

⁵ The numbers of respondents is generally considered a good number for testing IRT[11]

range [0.7, 1.312] for dichotomous items and [0.683, 1.287] for polytomous items, which are both within the typically accepted [0.5, 1.5] range [13].

Having shown the applicability of IRT to the cloud sharing scenario, we proceed to the evaluation of our framework.

5.2 Synthetic Datasets

In this section we detail our methodology for evaluating our framework with synthetic data, followed by the experimental results and discussion.

Methodology The context items in this dataset were generated by selecting a generic vocabulary with 5 fields per context. Each field of a context had 5 possible values for a total of 3125 possible contexts. From these contexts, we selected 200 ones at random. There are 500 sharers (or people) who share these items. In total, for each experiment we allocated 30000 sharing instances, each of which represents a data item (corresponding to the context item) shared by a certain person with another person at a specific time. The item to share at each instance is drawn according to a predetermined item distribution (zipf with exponent 2, or uniform, depending on the experiment). In our implementation, the distance (and hence similarity) between each pair of contexts is based on considering the hamming distance over their fields⁶. The people connections for sending data were modeled using two types of graphs: (i) small world (using the Watts-Strogatz model with a base degree of 2 and $\beta = 0.5$) and (ii) random (with average degree of 4). Our simulation is a discrete event based simulation, punctuated by sharing events. The person who instantiates a sharing event is selected randomly from the graph, weighted by her degree, so that people who have more neighbors share more items than those with less. The data receiver is selected randomly from the list of neighbors of the sender. Each person sends data at a time rate modeled by a Poisson process so that the time between her two sharing instances is exponentially distributed with an average of 3, 6, or 12 hours, depending on the experiment.

At each sharing instance, the context item’s *QuerySet* is sent according to our scheme. The server maintains clusters of contexts it receives, grouped according to a *similarity parameter* (whose value of 1 implies that each cluster’s contexts differ by one field from their cluster center, etc.). When the server receives a new context, it either maps it to an existing cluster or assigns it as the center of a new one. All the contexts of a certain cluster are assumed to have the same sensitivity. The server replies with all the sensitivities it knows for the clusters to which the contexts in the *QuerySet* were mapped. If the reply contains the requested item, this is considered as a *Hit*.

In the crowdsourcing phase, upon receiving new *Crowdsourcing Information (CI)* from a client, the server matches it to a cluster S and tries to compute the sensitivity for S if it is not yet computed. To achieve an acceptable sample for IRT, we require that (1) S has a minimum of 15 contexts with their policies,

⁶ System designers can use any similarity measure best suited for their needs, e.g., those dealing specifically with semantic similarity. However, that is beyond the scope of this work.

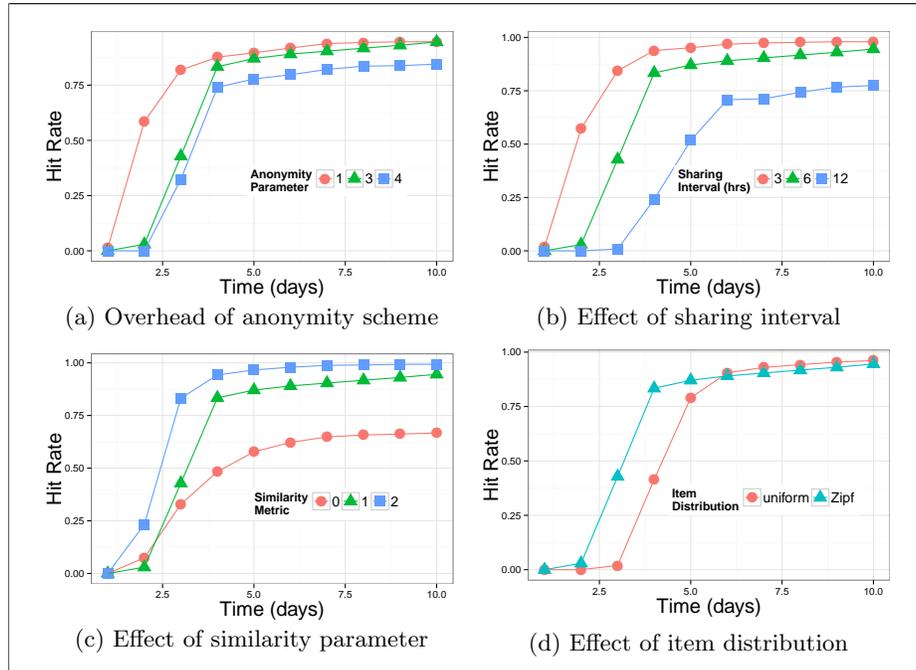


Fig. 6: Synthetic dataset graphs

(2) that there are 4 other clusters satisfying requirement 1, and (3) that each of these 4 clusters has at least 8 *CI*s by people who have also sent *CI*s appearing in S . The sensitivities are computed using the *marginal maximum likelihood estimation* technique. In all the experiments, unless otherwise stated, the default setting is a small world social network with zipf item distribution, six hours average sharing interval, and a similarity parameter of 1. In addition, the value for parameter r is equal to k , which is 3 by default. Hence, k is the anonymity parameter we use henceforth.

Results and Discussion Figure 6a shows the Hit Rate of users queries over time, where Hit Rate is defined as:

$$\text{HitRate} = \frac{\# \text{ of queried items with available sensitivity}}{\text{total \# of queried items}} \quad (2)$$

The Hit Rate is calculated per day unless otherwise specified. In Figure 6a we can see that the Hit Rate for anonymity parameter 3 is better than the Hit Rate for 4. As discussed earlier, anonymity parameter k implies that a *targetContext* for sensitivity must have appeared in k different anonymity subsets and that k different values for each field in the *targetContext* must be present in the *QuerySet*. The above conditions suggest that lower the anonymity parameter value, more *targetContexts* would be sent to the server for crowdsourcing, and thus more quickly would IRT be able to respond with sensitivity values.

The anonymity parameter 1 implies no anonymity at all. We plot this curve to see the ‘overhead’ of our K-anonymity scheme on top of the time required by IRT. Simply put, the curve for the anonymity parameter 1 represents the time it takes IRT to provide Hit Rates when there is no anonymity scheme in place. Thus the difference between the curves for anonymity parameters 1 and 3 represents the overhead of our anonymity scheme in terms of reduced Hit Rate. However, we see that the curve for 3 converges to the same Hit Rate as 1 in ten days time. This suggests that our anonymity scheme bootstraps quickly and does not pose significant overhead⁷

Figure 6b shows the Hit Rate with different sharing intervals in hours. An interval of 3 means that all users query for the sensitivity of an item every 3 hours on average. It can be seen from the Figure that initially, longer the interval, the slower the increase in the Hit Rate. This is most noticeable around the 5th day, when the Hit Rate with an interval 12 is still around 0.5 and lags significantly behind. Eventually, as the server collects more and more items, the Hit Rates of all sharing intervals converge to similar values.

Figure 6c shows the Hit Rate with different similarity parameters. Similarity parameter has been defined in Section 5.2 A similarity parameter of 0 signifies that there is no (zero) difference between two context items while calculating sensitivity⁸. Precisely, what this means is that to calculate the sensitivity of an item, IRT would require other items, which are exactly the same as this item, to have been shared with different policies. A similarity parameter 1 implies that two items that differ by a distance of 1 would be considered the same while 2 implies that items differ by a distance of 2 would be considered the same. This in turn implies that IRT would be able to more quickly calculate the sensitivity of an item (as opposed to case 0) since there would be more items which are considered the same. Thus we can see in Figure 6c that Hit Rate with similarity parameter 0 is the worst since IRT does not have enough items for calculation.

In Figure 6d, we investigate the effects of the ‘item distribution’ on the Hit Rate. By ‘item distribution’ we mean the distribution of the context items, i.e., the different contexts in which users share data. This is an important feature because different organizations and different systems would naturally have different item distribution. For our experiments, we use two different item distributions. One is the *zipf* distribution, which has been shown to be most common in social networks [14]. The other is the *random* distribution in which all context items are randomly distributed. A look at the Figure 6d reveals that a *zipf* distribution ‘bootstraps’ faster than a random distribution. The Hit Rate with random distribution lags behind zipf by a day, i.e., it reaches the same Hit Rate a day later, till the fifth day. We argue this is because, given a *zipf* distribution, users share more similar items, and thus the crowdsourcing is more effective, and IRT is able to calculate the sensitivity of items quickly. Given a random distribution, it takes more time for IRT to accumulate enough similar items for the calculation

⁷ This overhead can be further reduced through bootstrapping the system with initial data collected from surveys, thus increasing the Hit Rate at the beginning.

⁸ This was the case for example in the experiments for validating IRT in Section 5.1

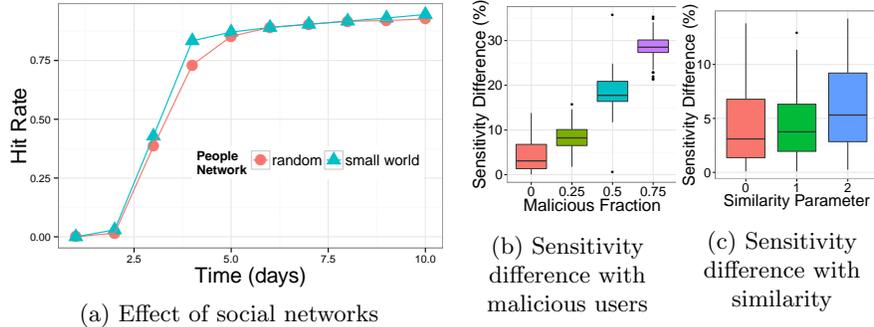


Fig. 7: Hit rate and sensitivity difference under various conditions

of sensitivity. However, as the times goes by and more items are accumulated, both random and zipf converge to around the same values.

In Figure 7a we observe the effect of changing the underlying social network. We use two graphs for the social network structure: small world and random. These affect the sharing patterns of the users. We see that the effect of the underlying graphs on the Hit Rate is not significant and both lead to similar values, with the small world doing slightly better than the random network.

Finally, Figures 7b and 7c show the effect of malicious users and changing similarity parameters on the sensitivity values. For this particular set of experiments, we begin by assigning different sensitivity values to the items and also different attitudes to the users. As the experiment runs, the policy of the users on sharing the items is dictated by their attitude and the item sensitivity given at the beginning. The sensitivity of the items is then calculated by our scheme. We then measure the absolute difference between the actual sensitivity of the items and the calculated sensitivity. Ideally, there should be no significant difference between the actual sensitivity and the calculated sensitivity. However, differences could arise under certain conditions. The first condition is the presence of *malicious users*. A malicious user sends random policies for items, i.e., she does not have a fixed attitude but rather a random and unpredictable one.

Figure 7b shows the effect of such malicious users on our scheme. The figure shows the box plots for each item’s normalized sensitivity difference in terms of percentage. We observe that when there are no malicious users, the difference is pretty low (in the range [2%, 6%]), with most items calculated sensitivity very near the actual sensitivity (the individual dots represent the outliers). This keeps getting progressively worse as the proportion of malicious users increases. Finally, with a fraction of 0.75 malicious users, most of the items’ calculated sensitivity differs by as much as 30% from the actual sensitivity.

In Figure 7c, we see that the effect of different similarity parameters on the calculated sensitivity. We can see that, with similarity parameters 0 and 1, the difference between actual and calculated sensitivity is very low. The reader will recall that similarity parameter 0 means that two items would only be grouped together if they are identical. Therefore, when IRT calculates sensitivity value of an item, it does so on the basis of other identical items for which it has received

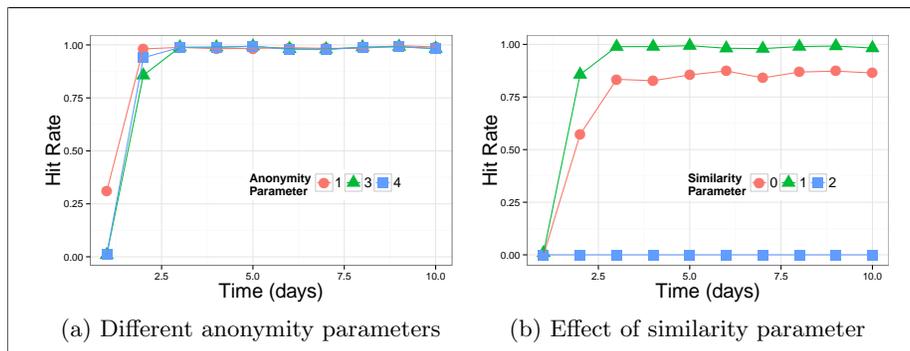


Fig. 8: Enron dataset graphs

policies from different users. Thus the calculated sensitivity value would be in high agreement with actual sensitivity. With increasing similarity parameter values, the system would group together items which are not identical, therefore sacrificing accuracy in sensitivity calculation. We observe that the difference of actual and calculated sensitivity with similarity parameter 2 is greater than 0 and 1. However, as we discussed while explaining the results of Figure 6c, a higher value for the similarity parameter signifies a better Hit Ratio. Therefore, we discover that there is a *tradeoff* between accuracy of calculated sensitivity and Hit Rate, as far as similarity parameter is concerned.

5.3 Enron Experiments

We want to evaluate our scheme in a realistic setting. However, as there is no dataset of users sharing activities in the cloud that is publicly available, we use the Enron email dataset. Sharing data of various types with certain entities in the cloud is analogous to sharing attachments via emails. Specifically, what we get from this dataset, is a variety of items (hence variety of contexts in which real people share these items with others) and also the level of trust that they have in each other. We explain these points as well as our data extraction and analysis methodology below.

Methodology The dataset was obtained from (<http://info.nuix.com/Enron.html>) in the form of 130 personal storage folders (pst). It was processed using the PST File Format SDK⁹ and the MIME++ toolkit¹⁰. We only considered emails with attachments, whose metadata was extracted using the GNU Libextractor library¹¹. Precisely, the main metadata we extracted from files is: (revision history, last saved by, resource type, file type, author name, and creator). We then collected all the email addresses mentioned in the dataset and grouped the ones corresponding to the same person, based on the patterns of occurrence of email aliases. Next, the emails of each person were used to obtain the list of companies she is affiliated with according to the email domain, filtering out public

⁹ <http://pstsdk.codeplex.com/>

¹⁰ <http://www.hunnysoft.com/mimepp/>

¹¹ <http://www.gnu.org/software/libextractor/>

email services (e.g. AOL, Yahoo). We matched all the processed metadata with a specific vocabulary we created for the Enron Dataset. In total the obtained dataset contained 2510 people sharing 184 distinct contexts over 19415 sharing instances. Moreover, for each file sender, we calculated a measure of the trust associated with each receiver based on the frequency of emails exchanged with her. The trust value $T(i, j) = F(i, j)/Max(i)$, where $F(i, j)$ is the number of emails sent from user i to user j , and $Max(i)$ is the maximum number of emails sent by user i to any receiver. In our experiments, the policies we associate with each sending event are dictated by this degree of trust by the sender in the receiver. We use a similar timing scale as the synthetic experiments, where each person shares all his items in the sequence the emails were originally sent but with a rate modeled as a Poisson process.

Results and Discussion Figure 8a shows the Hit Rate of users queries over time, where Hit Rate is the same as defined in Equation 2. The graph is over a period of 10 days. We can see that with anonymity parameter 1, i.e. with no anonymity scheme in place, the Hit Rate jumps very quickly. However, anonymity parameter 3 and 4 eventually catch up and all the curves show a Hit Rate of 1 by the third day. We argue that this improvement in Hit Rate over the case of synthetic experiments (see Figure 6a) is because the sharing contexts in the Enron dataset are not diverse and more similar items are collected faster, thus leading to an increase in the Hit Rate.

In Figure 8b we can see that with the similarity parameter equal to 2, the Hit Rate remains at 0 consistently. Our investigation into this reveals to us the reason behind this strange result. We discover that the context items shared in the Enron dataset are not very diverse. Hence, having a similarity value of 2 means that most items are clustered together since most items in the Enron dataset differ from each other by at most 2 fields. As most items are clustered in very few clusters, this means that IRT is not able to work since it does not find enough different items to sensitivity calculation. The number of different items that IRT requires for working differs on the implementation being used. Our implementation requires that there must be at least 5 different items for IRT to work. In case of similarity 2, these clusters are not available.

However, with similarity 1, enough clusters are found and this in turn implies that IRT would be able to more quickly calculate the sensitivity of an item (as opposed to case 0) since there would be more items which are considered the same. Therefore, similarity 1 shows better Hit Rate than similarity 0.

These results suggest that using IRT in a scenario where most people share similar items, the similarity parameter should be low. However, we note that the Enron dataset, being an email dataset, does not have the same diversity as would be available in a cloud setting. Thus, we have shown the feasibility of our scheme using synthetic and empirical data.

6 Related Work

One of the relevant attempts at privacy risk estimation was in the field of social networks. Liu and Terzi [7] used IRT in quantifying the privacy risk of exposing user profile items on Facebook (e.g. birthday, political affiliation, etc.). Kosinski

et al. [8] also modeled the process of information disclosure in Facebook using IRT and found correlations between disclosure scores and personality traits. Our work is distinct from these in that IRT is shown to apply in the case of privacy aware sharing in the cloud and in that it utilizes context extraction to work with any data file, without being limited to a predefined set of profile items.

The concerns about the privacy of user data in the cloud were confirmed by Ion et al. [15] through surveys highlighting the users' beliefs about the intrinsic insecurity of the cloud. A client-centric system was also developed by Ion et al. [16], enabling users to share text and image files on web-based platforms while keeping the data encrypted and hiding the fact that confidential data has been exchanged from a casual observer. From one angle, our work is complementary to theirs as we design the framework to help users decide on what data to keep confidential. From another perspective, our work is distinct as we allow multiple levels of privacy policies that can be controlled by the user.

The work by Garg et al. [17] highlights the *peer produced privacy* paradigm, which treats privacy as a community good and considers individuals who share the risk of information sharing. The authors argue that such an approach can result in more socially optimal privacy decisions. Our work shares similar motivations, among which are the suboptimal privacy decisions taken by average users and the inability of users to keep track of the changing contextual factors affecting privacy. We envision that extensions of our work can exploit this paradigm for producing socially optimal privacy policy recommendations.

The concept of contextual privacy has also received significant attention recently. Nissenbaum's work (e.g. [18]) was one of the notable contributions that called for articulating context-based rules and expectations and to embed some of them in law. Several works have developed context-aware systems for privacy preservation in scenarios like sensor data sharing [19] and mobile social networks [20].

7 Future Work

In this paper, we have provided a novel framework for preserving the privacy of data shared to the cloud. One of the future directions we are planning to investigate is further improving the privacy guarantee of our scheme to resist probabilistic attacks by a server trying to link a context with a sharing operation. Moreover, we are currently investigating alternative techniques to IRT, such as *Bayesian Networks*, that can also serve to model people's privacy aware sharing. Also, developing techniques for recommending privacy policies to users is one of the main pillars of the final system. As of writing, we are working on developing the first prototype for our system, which will be released in the near future.

Acknowledgment The research leading to these results has received funding from the EU in the context of the project *CloudSpaces*: Open Service Platform for the Next Generation of Personal clouds (FP7-317555).

References

1. Greenwald, G., MacAskill, E.: NSA Prism program taps in to user data of Apple, Google and others. *The Guardian* **7**(6) (2013) 1–43

2. Gentry, C.: A fully homomorphic encryption scheme. PhD thesis, Stanford University (2009)
3. Van Dijk, M., Juels, A.: On the impossibility of cryptography alone for privacy-preserving cloud computing. In: Proceedings of the 5th USENIX conference on Hot topics in security. (2010) 1–8
4. Protiviti: Knowing how – and where – your confidential data is classified and managed. Technical report, Protiviti Inc. (2013)
5. Sweeney, L.: k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* **10**(05) (2002) 557–570
6. Baker, F.B.: The basics of item response theory. ERIC (2001)
7. Liu, K., Terzi, E.: A framework for computing the privacy scores of users in online social networks. *ACM Transactions on Knowledge Discovery from Data* **5**(1) (2010) 6
8. Quercia, D., Casas, D.L., Pesce, J.P., Stillwell, D., Kosinski, M., Almeida, V., Crowcroft, J.: Facebook and privacy: The balancing act of personality, gender, and relationship currency. In: International AAAI Conference on Weblogs and Social Media. (2012)
9. Reeve, B.B., Fayers, P.: Applying item response theory modeling for evaluating questionnaire item and scale properties. *Assessing quality of life in clinical trials: methods of practice* **2** (2005) 55–73
10. Nering, M.L., Ostini, R.: Handbook of polytomous item response theory models. Taylor & Francis (2011)
11. Linacre, J.M.: Sample size and item calibration stability. *Rasch measurement transactions* **7**(4) (1994) 328
12. Mair, P., Hatzinger, R.: Extended rasch modeling: The erm package for the application of irt models in r. *Journal of Statistical Software* **20**(9) (2007) 1–20
13. De Ayala, R.J.: Theory and practice of item response theory. Guilford Publications (2009)
14. Lewis, K., Kaufman, J., Gonzalez, M., Wimmer, A., Christakis, N.: Tastes, ties, and time: A new social network dataset using facebook.com. *Social networks* **30**(4) (2008) 330–342
15. Ion, I., Sachdeva, N., Kumaraguru, P., Čapkun, S.: Home is safer than the cloud!: Privacy concerns for consumer cloud storage. In: Proceedings of the Seventh Symposium on Usable Privacy and Security. (2011) 13:1–13:20
16. Ion, I., Beato, F., Čapkun, S., Preneel, B., Langheinrich, M.: For some eyes only: Protecting online information sharing. In: Proceedings of the Third ACM Conference on Data and Application Security and Privacy. (2013) 1–12
17. Garg, V., Patil, S., Kapadia, A., Camp, L.J.: Peer-produced privacy protection. In: IEEE International Symposium on Technology and Society. (2013) 147–154
18. Nissenbaum, H.: A contextual approach to privacy online. *Daedalus* **140**(4) (2011) 32–48
19. Pallapa, G., Di Francesco, M., Das, S.K.: Adaptive and context-aware privacy preservation schemes exploiting user interactions in pervasive environments. In: IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks. (2012) 1–6
20. Bilogrevic, I., Huguenin, K., Agir, B., Jadliwala, M., Hubaux, J.P.: Adaptive information-sharing for privacy-aware mobile social networks. In: Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing. (2013) 657–666