Analysing and Evaluating Syntactic Privacy Games via a Recommender Systems Case Study

Robin Ankele
Department of Computer Science
University of Oxford
Oxford, United Kingdom
robin.ankele@cs.ox.ac.uk

Andrew Simpson
Department of Computer Science
University of Oxford
Oxford, United Kingdom
andrew.simpson@cs.ox.ac.uk

***Abstract*—Establishing an accurate and commonly accepted definition of privacy remains something of a 'holy grail' for privacy researchers. The task seems impossible due to the diversity, multi-facetedness and multi-dimensionality of privacy. Further, privacy is often context-sensitive or data-dependent. In recent years, the probabilistic notion of differential privacy has emerged as a popular measure for systems that support interactive data release. Consequently, there exists a portfolio of mechanisms to achieve differential privacy. A complementary approach is to represent syntactic privacy notions such as anonymity, unlinkability, pseudonymity and unobservablility in the form of games. Via these games, the understanding of, and relationship between, privacy notions is clarified; further, an unambiguous understanding of adversarial actions is given. Representing privacy notions as games is aimed to invoke a 'different' perspective of visualising and abstracting privacy notions. Yet, without any practical context, these notions can be incomprehensible to system designers and software developers. In this paper, we provide means to combine, enhance and extend these theoretical contributions by showing that the game-based definitions have the potential to interconnect privacy implications and to reason about privacy properties. We motivate our analysis through a case study based on recommender systems — systems in which the proposed privacy games have a natural fit. Taken together, our contributions aim to promote a better understanding of the privacy notions and games, and increase their usability, flexibility and applicability for non-experts.**

***Index Terms*—Privacy Notions, Privacy Games, Recommender Systems, Release Privacy**

1. Introduction

Establishing an accurate and commonly accepted definition of privacy has proven to be a hard task. By its very nature, privacy is diverse, multi-faceted and multi-dimensional. As such, a precise definition of privacy may not be applicable in different application contexts (nor is there any realistic prospect of there ever being a universal definition).

Within recent years, the notion of differential privacy [1], [2] has emerged as a popular means of reasoning about privacy in data release scenarios. Consequently, a portfolio of various mechanisms to achieve differential privacy has been established: from generic and data-independent mechanisms to mechanisms associated with a particular task.

In an alternative approach[1], Bohli and Pashalidis [3] define various privacy notions, which are associated with the privacy properties that a system can inherit. These privacy notions are defined on properties of the system's release: for example, a system achieves strong anonymity if an adversary is not able to associate any relationship between the outcome element and a user identifier (pertaining to the outcome element).

A drawback is that these notions can be elusive, and prone to misunderstanding by system designers and software developers. In [4] the present authors showed how these notions can be represented in the form of games — to enhance the understanding of, and characterise the relationships that hold between, these different notions. Via these games, the adversarial actions to break a notion are clearly defined.

The presentation of the games defined in [4] was largely theoretical: no case studies, use cases or examples were used to demonstrate applicability. The case study described in this paper[2] has been developed to help bridge this gap: we use the presented case study to motivate and validate the analysis of the privacy notion and games — thus, helping to demonstrate the applicability of approaches such as described in [3] and [4].

The kind of tasks of interest to us pertain to privacy-preserving data mining. Data mining is the computational process of finding and representing patterns in large data sets. In this paper, we are particular interested in two classes of data mining: *clustering* and *classification*. Clustering is the task of grouping sets of elements: elements in a group are more similar to each other than those appearing in other groups. In the context of machine learning, it pertains to unsupervised learning methods operating on 'un-labeled' data. A clustering algorithm increases its accuracy by learning from training data. Classification is

1. Differential privacy [1], [2] is a property of the release algorithm and, as such, independent of the processed data. The level of privacy guarantees / data utility that can be achieved depends on the sensitivity of the query function. Differential privacy is an inherent probabilistic approach. Alternatively, syntactic privacy notions (including our privacy games) are dependent on the data. Concretely, privacy notions are defined on the relations between data elements. The level of privacy guarantees depends on how a release algorithm can obfuscate these relationships.

2. This paper is an extended version of an abstract presented at PST 2018 [5]

the task of assigning new observations to a pre-defined set of classes; each class portrays a set of different properties. Classification algorithms pertain to supervised learning methods operating on clearly 'labeled' data. During the training phase, as a lazy learner, it stores data, while making decisions during the operational phase. Our case study is based upon recommender systems [6], [7], with deployment of both of these classes of data mining. The case study was chosen because it fits within the application scenario of data mining. Further, the privacy implications within a recommender system are easy to understand, and the notions and games that are of interest to us are well reflected within such a context.

In addition to the case study, we study the notions and games theoretically. We show limitations and give probabilistic bounds on the likelihood of an adversary breaking an associated notion. Such knowledge is essential to system designers who are concerned with choosing a privacy notion or privacy game for their own systems.

To summarise, we claim the following contributions. First, we present a case study based on recommender systems that uses clustering and classification methods and has a privacy-preservation. Second, we show theoretical limitations and probabilistic bounds of the notions and games. This paper extends the abstract [5] presented at PST 2018, by an extensive analysis of the privacy notions and privacy games (Section 5) and our visual representation of the notions and the system model given in Section 6. In addition, the contributions in this paper extend our previous work [4], which saw mainly the abstraction of the privacy notions via games by a first approach to utilise these game based definitions in a case study to reason and argue about privacy properties.

We note that, while the presented notions and games in this paper can capture a wide range of privacy abstractions, not all issues are characterised by these definitions. As such, some restrictions are inherent from both the system model and the adversary model. The approach is simply an alternative, complementary way of thinking about a pressing problem.

The remainder of this paper is organised as follows. In Section 2, preliminary background information is presented; in detail, we outline the underlying system model, summarise the notions presented by Bohli and Pashalidis [3] and games presented by Ankele and Simpson [4]. In Section 3, the case study that is used to motivate and to validate our contribution is illustrated. Section 4 presents our mapping of the privacy notions and games to our system model and case study. In Section 5, we present the results of our theoretical analysis. Section 6 presents an illustration of the privacy notions in conjunction with the system model. Finally, we conclude in Section 7 and, importantly, outline potential areas of future work.

2. Background

2.1. System Model

A system, denoted by ϕ, takes a finite set of input parameters, denoted as $\alpha_i \in \vec{\alpha}$, and releases outcome elements, $e_i \in \vec{e}$, together with a single (aggregated) value, β. For each input element, α_i, as well as for each outcome element, e_i, a user identifier, u_i, is assigned. These values are drawn either from an input space, $\mathcal{I}$, or from an output space, $\mathcal{O}$.

Users interact with systems via interfaces — for example, an interface, $\texttt{input}(\cdot,\cdot,\cdot)$, to process inputs to the system and an interface, $\texttt{view}^{\pi_q}(\cdot)$, to receive the system release. A system, ϕ, may consist of a single party, p, or a set of parties, $p_i \in \vec{p}$, in order to store and process data. Essentially, a system is a higher level abstraction of a party or set of parties; the parties are the engines to store, represent and operate on the data.

Within a system, a set of transformation functions, denoted by $\vec{\pi_q}$, map a value, α_i (or a set of values, $\alpha_i \in \vec{\alpha}$), to an outcome value, e_i, as well as a single aggregated value, β. The length of the outcome parameters is not necessarily the same as the length of the input parameters. (Formally, $|\vec{\alpha}| = |\vec{e}|$ or $|\vec{\alpha}| \neq |\vec{e}|$; in particular, such a mapping may be non-injective.[3]) Furthermore, transformation functions may be composite thus: $\vec{\pi_q} = \pi_{q,1} \circ \pi_{q,2} \circ \cdots \circ \pi_{q,n}$, with $(\vec{e}, \beta) = \vec{\pi_q}(\vec{\alpha})$ and any $\pi_{q,i}$ takes as input the released values of $\pi_{q,i-1}$ and inputs to $\pi_{q,1}$ are $\alpha_i \in \vec{\alpha}$ and release elements of $\pi_{q,n}$ are $(\vec{e}, \beta)$. An illustration of such a mapping is given in Figure 2.

In order to abstract privacy properties for systems, Bohli and Pashalidis [3] have, for their privacy notions, and, subsequently, the present authors [4] for their privacy games, defined a function $f : e_i \rightarrow u_i$, which associates the user identifier, u_i, with the released elements, e_i. This association is seen to be *sensitive* if learned by an adversary, $\mathcal{A}$. Using privacy mechanisms as part of the set of transformation function, $\vec{\pi_q}$, may prevent such information gain. In addition, the function $f^\star : \alpha_i \rightarrow u_i$ maps an input element, α_i, to the pertaining user identifier, u_i. Depending on the adversarial model, $\mathcal{A}$ may learn this relationship.

Besides the definition of the mapping functions f and $f^\star$, we additionally inherit the definition of the sets U_f, Q_f and P_f (as stated in [4], and originally adapted from [3]) as follows:

Definition 1. *(Participant Set):* $U_f = \{f(e_i) : e_i \in \vec{e}\}$ denotes the set of participants in ϕ. Each element, $e_i \in \vec{e}$, is uniquely associated with a user identifier, $f(e_i) = u_i$.

Definition 2. *(Usage Frequency Set):* $Q_f = \{(u_i, \#u_i) : u_i \in U_f\}$, with $\#u_i = |e_i \in \vec{e} : f(e_i) = u_i|$, denotes the relation of the number of elements, $e_i \in \vec{e}$, with which each user, u_i, is associated. In other words, $\#u_i$ is the number of elements that user u_i has contributed to ϕ.

Definition 3. *(Linking Relation):* $P_f = \{\vec{e}_{u_1}, \vec{e}_{u_2}, .., \vec{e}_{u_{|U_f|}}\} \vdash \vec{e}$, denotes the linking relation of elements $e_i \in \vec{e}$, which pertain to a certain user, u_i. In other words, $\vec{e}$ is partitioned into disjoint subsets, $\vec{e}_{u_i}$, where $\vec{e}_{u_i}$ consists of all e_i for user u_i. Precisely, it holds that $\forall e_i \in \vec{e}_{u_i} : f(e_i) = u_i$.

Figure 1 further illustrates the abstraction of the mapping function f. Moreover, it indicates how the previously

3. We do not require the mapping to be surjective, since this would mean that every 'possible' element of the codomain is mapped to by at least one element of the domain. Yet, we require that every 'actual' output element must be mappable (through inversion of the privacy-preserving function) to an input element.

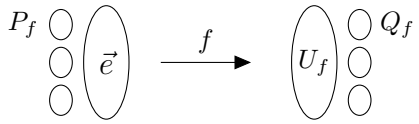

Figure 1: Abstraction of the mapping function $f : e_i \rightarrow u_i$: each element $e_i \in \vec{e}$ maps to a unique user identifier $u_i \in U_f$. P_f partitions $\vec{e}$ into disjoint subsets; likewise, Q_f partitions U_f into disjoint subsets.

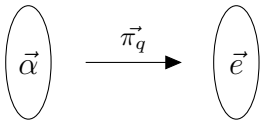

Figure 2: Abstraction of the transformation function $\vec{\pi_q} : \alpha_i \rightarrow e_i$: elements $\alpha_i \in \vec{\alpha}$ map to output elements $e_i \in \vec{e}$. $\mathcal{A}$ may know $f^\star(\alpha_i) = u_i$, but must not learn $f(e_i) = u_i$.

defined sets U_f, Q_f and P_f are associated to f.

2.2. Privacy Notions and Privacy Games

In order to model or abstract certain properties of a system or, data, released by a system, privacy notions are defined. Such privacy notions reflect these properties or requirements. For example, Bohli and Pashalidis [3] present the notions of anonymity, participation hiding, unlinkability and pseudonymity. These notions (and an additional notion of unobservability) are abstracted in the form of privacy games in [4], to ease the understanding of said notions, and to characterise the relationships between them.

Privacy abstractions have been studied in many fields with the aim to foster and improve the understanding and reasoning of, and between, these notions. Alternatively, game-based security notions have been widely used in the cryptography community to provide simpler and more easily verifiable proofs. For this technique [8] notions are formulated in games illustrating the necessary actions for an adversary to break said notion. Examples include [8], [9]. Moreover, such games are also widely used to reason about privacy properties (for example, within anonymous authentication protocols [10], [11]). In this context, games abstract notions of unforgeability, seclusiveness, anonymity or unlinkability in pseudonymous signatures schemes [12] or attribute-based credential schemes [13]. Privacy games enjoy a wide popularity and their simplicity help non-experts to understand some otherwise complex mathematical definitions. It is of these reasons, why we selected and aim to improve on this approach.

Each privacy notion embodies its own restrictions or requirements on the released data from a system. Table 1 summarises the notions we consider in this paper. The notions are listed with respect to a partial order dependent on the knowledge of an adversary, $\mathcal{A}$ (and its access to oracles).

In brief[4], the major notions can be stated as follows.

(a) For **anonymity**, an adversary should not be able to learn any interesting relationship between single or groups of outcome elements (especially the user identifier) with regards to the input elements.
(b) For **participation hiding**, an adversary should not be able to link an outcome element to a specific user identifier.
(c) In the case of **unlinkability**, an adversary should not be able to link released elements pertaining to the same user identifier.
(d) For **pseudonymity**, we may allow an adversary to link element groups with each other; however, we aim to prevent her from learning the actual user identifier, though allowing her to assign that group with a pseudonym.
(e) For **unobservability**, an adversary may not be able to distinguish a system invocation from a 'fake' system invocation that releases a 'random' outcome element.

In [4] we associated to each privacy notion a game, which an adversary must win in order to break the privacy notion. In the game-based definitions, an adversary has access to a number of interfaces, $\mathcal{I}$. Within the games, interfaces are represented as procedures, denoted as **proc**. Each game consists of a limited number of such procedures, which outline the necessary steps in the form of an algorithm that an adversary needs to perform in order to break the associated notion. Some of these procedures may be executed an arbitrary amount of times, denoted by $\mathbf{proc}_{\circlearrowleft}$, while others may be executed only once.

Importantly, an adversary may only be able to query each of the various `validate` interfaces once. Thereafter, the dataset needs to be destroyed to maintain a guarantee of privacy — to prevent $\mathcal{A}$ from submitting, in a brute-force fashion, queries to the validate interface in the hope of hitting the correct result. In real systems this may not always be applicable. As such, in a more restricted setting it can be argued that privacy may only be guaranteed as long as an adversary is not exceeding a previously defined number of queries to the `validate` interface.

An adversary, $\mathcal{A}$, 'breaks'[5] a privacy notion if she is able to come up with a 'strategy' (in the literature often referred to as a 'distinguisher') such that, for multiple executions of a privacy game, the associated game always evaluates to true with a non-negligible probability. Here, 'non-negligibility' is defined with respect to the number of users / elements in system ϕ (*i.e.* the anonymity set). Thus, we require $|U_f|$ and $|\vec{e}|$ to be large (*i.e.* computationally infeasible to perform random guesses), as otherwise an adversary $\mathcal{A}$ would always be able to succeed in 'breaking' a notion with a non-negligible success probability. The success probability of $\mathcal{A}$ is given by $\mathbf{Succ}(\mathcal{A}) = Pr[G_\star \Rightarrow \texttt{true}]$ where $\star \in \{\texttt{SA}, \texttt{PH}, \texttt{SU}, \texttt{WU}, \texttt{PS}, \texttt{AN}, \texttt{WA}, \texttt{UO}\}$[6].

In order to ensure correctness, we assume that an adversary is semi-honest and follows the instructions of the game honestly — to execute instructions top down and in order, and to only submit values to the interfaces that

4. For a more detailed introduction the reader is referred to [3] or [4].

5. A system, ϕ, achieves or gives guarantees conveyed by a privacy notion. With 'break' we mean that an adversary is able to win a game (*i.e.* the game returns true) and, as such, ϕ does not retain these guarantees anymore.

6. Here, the notation follows the notions of strong anonymity (`SA`), participation hiding (`PH`), strong unlinkability (`SU`), weak unlinkability (`WU`), pseudonymity (`PS`), anonymity (`AN`), weak anonymity (`WA`), and unobservability (`UO`). For a precise definition of each notion, the reader is referred to [3].

TABLE 1: Summary of the notions for privacy considered in our privacy model.

Privacy Notion	Category⋆	Learning Goal⋆⋆	Knowledge of $\mathcal{A}^{\dagger}$
SA	single elements	$f(e_x) = u_x$	$-$
PH		$f(e_x) = u_x$	$\lvert U_f \rvert$
SU	multiple elements	$f(e_x) = f(e_y)$	U_f
WU		$f(e_x) = f(e_y)$	U_f, Q_f
PS	element groups	$f(\vec{e}_{u_x}) = u_x$	P_f
AN		$f(\vec{e}_{u_x}) = u_x$	U_f, P_f
WA		$f(\vec{e}_{u_x}) = u_x$	U_f, Q_f, P_f
UO	system behaviour	validity of $\vec{e}, \beta$	U_f, Q_f, P_f

⋆ The notions are grouped regarding similar objectives in distinct categories.

⋆⋆ The learning goal indicates what an adversary is aiming to achieve for each category.

† The knowledge of $\mathcal{A}$ indicates access to certain interfaces of ϕ depending on the privacy notion under question.

it previously obtained from a system output (except in the case of the access to the input interface). A more detailed discussion of the capabilities of an adversary is given in Section 5.2. In [4] we also define a inheritance relationship between games, denoted by $G_x \vDash G_y$, which allows a game, G_x, in addition to calling its own procedures, to also call the procedures of game G_y.

3. A Case Study: Recommender Systems

With the rapid explosion of new developments and technologies for web-based services in recent years, there has been, consequently, an increase in the complexity and choice a user faces when accessing these services. This significant growth of services has the potential to lead users to struggle to find the information they are looking for and can make the selection of a product difficult and time-consuming. The deployment of a recommender system has proven to be an efficient method to tackle this problem. The suggestion of similar products may allow a user to make a choice from a wide product range. Hence, it can reduce the information overload a user has to deal with while searching for similar items. Such systems also help to cross-sell additional products that a user might not have thought of, but may be interested in buying.

For a recommender system to work efficiently, it requires content-based methods or collaborative methods, the analysis of users' preferences, and users' buying history. While several kinds of recommendation methods have been proposed, within this paper we focus on *(a) content-based recommendations* and *(b) collaborative recommendations*. The former method generates recommendations based on similar items; the latter generates recommendations based on user collaborations or similar user behaviour[7].

In order to make recommendations that are both accurate and diverse, it is first necessary to collect a substantial amount of user data and purchase history. Furthermore, most recommender systems collect users' preferences. Such collections raise concerns about an individual's privacy (or the privacy of a group / cluster of users). Examples include: vulnerabilities in collaborative filtering based recommendation algorithms (*e.g.* the k-nearest neighbour algorithm) shown by Calandrino *et al.* [14]; privacy concerns for users that rate elements across disjoint domains shown by Ramakrishnan *et al.* [15]; and privacy concerns through user control shown by Zhang *et al.* [16].

7. We know, for example, that users in a specific class or cluster will have similar interests. Thus, it is natural to recommend products bought by users in a cluster to others in the same cluster.

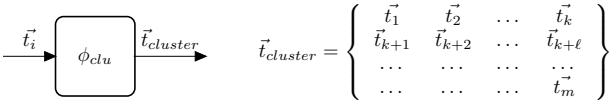

Figure 3: A recommender system deploying content-based clustering: it takes transactions $\vec{t_i}$ as input and releases a cluster $\vec{t}_{cluster}$ of similar transactions. It is the case that $\vec{t_i} \in \vec{T}$ and $\vec{t}_{cluster} \subseteq \vec{T}$, where $\vec{T}$ denotes the set of all transactions and $m = |\vec{t}_{cluster}|$. Each $\vec{t_i}$ must have a unique member $\vec{t_i}.user$ and $\vec{t_i}.product$.

3.1. Content-based Clustering

We consider a system that recommends products based on previous transactions. As such, the system takes as input user transactions and releases clusters of similar transactions. Using these clusters, we can recommend products to other users pertaining to the same cluster.

Examples of clusters in this context include (but are not limited to): transactions below a certain price threshold, transactions within a certain store, and transactions that involve a certain category of products. Figure 3 illustrates an example cluster.

To clearly represent and illustrate our recommendation method, we define some properties pertaining to transactions. Thus, each transaction has relationships to certain properties in the form of a function. The properties must include a *unique* user identifier and a product identifier, and may also include a price, a rating or a store identifier.

Our recommendation method releases clusters as an outcome. Within a cluster, multiple transactions may pertain to the same user; however, each single transaction must pertain to exactly one user. For each product a user buys, a mapping associated with the transaction is stored in the recommender system. After a certain threshold of transactions, the system may be queried to release a cluster of transactions. Using these clusters, recommendations can be computed.

The privacy implications of such a recommendation method are numerous. For example, any analyst may be

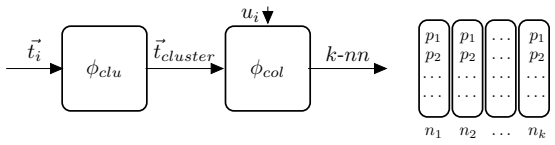

Figure 4: A recommender system deploying collaborative-filtering based classification: in the first step transactions $\vec{t_i}$ are clusterised into clusters $\vec{t}_{cluster}$; in the second step, for a user u_i, the products p_i of all k-nearest neighbours $(n_1, \ldots, n_k)$ are released. Recommendations can be generated via a majority vote over the products p_i among all neighbouring users.

permitted to learn a cluster of transactions. In contrast, if we consider a potential malicious analyst (denoted by $\mathcal{A}$): $\mathcal{A}$ must not be able to learn the user who conducted the transaction; further, $\mathcal{A}$ must not be able to link transactions pertaining to the same user. We apply the privacy notions and games on the released clusters.

3.2. Collaborative-filtering based Classification

We motivate our second collaborative-filtering (CF) based classification approach by the k-nearest neighbour attack. That is to say, with our approach, we aim to illustrate a CF-based classification algorithm that is able to withstand this kind of attack. As such, we consider a more complex, two-step method[8]: classification as a second step after clusterisation. As such, we reuse the released clusters from the recommendation algorithm described previously.

As a first step, we generate the transaction clusters. In the second step, we use these clusters as a 'classifier' — in order to assign (*i.e.* classify) if a new user added to the system pertains to a certain class of k-nearest neighbours. As the last step, we use a simple majority function over the k-nearest neighbours to recommend products to the users.

Figure 4 illustrates this multi-step recommendation method. As in the system described previously, our multi-step recommendation system remains filled with transactions. We release clusters as a first step and k-nearest neighbours as the second step. From the k-nearest neighbours, we release recommendations of single products. The privacy implications of this system are as follows: the clustering implications remain the same; for the classification step, we want to defy the k-nearest neighbour attack. We argue that with our method we do not generate the k-nearest neighbours from users who bought similar products, but via users that performed similar transactions — transactions within the same transaction cluster. As such, knowledge of products bought by a user does not reveal the k-nearest neighbours. Yet, we generate the k-nearest neighbours using the publicly released clusters. However, since we require that an adversary must not be able to learn the user identifier associated with a release transaction within any transaction cluster, revealing the clusters is fine. The privacy notions and games can be applied to the released k-nearest neighbours in order to reason about privacy.

8. To understand our reasoning of this two step recommendation, we remind the reader of the k-nn attack [14]: under the assumption that an adversary knows a certain subset of products bought by a user, an adversary generates k fake users. All of these fake users are then assigned the same subset of products as the target user. Thus, when calculating the k-nearest neighbours for the targeted user all of these fake users are included. By querying for a recommendation all the products bought by the targeted user are revealed — a severe break in the targeted user's privacy.

4. Concrete Mapping of Privacy Games to the Recommender Case Study

We now provide a formal discussion of the case study with regards to the privacy notions and games presented by Bohli and Pashalidis [3] and by the present authors [4]. We first provide a mapping to the underlying system model of these contributions. From this, we argue which privacy notions are achievable in the context of interest by analysing the games associated with each privacy notion.

4.1. Overview

In order to directly apply the privacy notions and games, the underlying systems must release information in a specific way. In particular, a system must release multiple outcome elements; each outcome element must be associated to a user identifier. Furthermore, outcome elements may stand in a relationship with other outcome elements. Such relationships are represented by a function, denoted by f — as illustrated in Figure 1.

In general, we consider the following release scenario: a system obtains batches of information from several parties and applies any data transformation function on the collected data. Before its release, a privacy-preserving function is applied. A system designer may now use the notions and games of [4] to give consideration to privacy properties with respect to this 'privacy-enhanced' release: the release of a system may be $\star$-secure, with $\star \in \{$`SA`, `PH`, `SU`, `WU`, `PS`, `AN`, `WA`, `UO`$\}$, if no adversary is able to win any of the games with more than a non-negligible probability.

4.2. Content-based Clustering

Consider a content-based clustering recommendation system, ϕ_{clu}. We have transactions, $\vec{t_i}$, as input values, $\vec{\alpha_i}$. A transaction is indeed a vector of multiple input values (*i.e.* a transaction has properties such as price and user identifier); however, we treat each single transaction as a single input value, α_i. In the next step, the recommendation system, ϕ_{clu}, performs a set of transformation functions, $\vec{\pi_q}$ — the clusterisation method. As a result of the clusterisation, ϕ_{clu} releases a vector of transactions: cluster $\vec{t}_{cluster}$, denoted by the output value vector, $\vec{e_i}$, for release query, q_i.

As discussed previously, we require that each transaction $\vec{t_i}$ maps to a unique user identifier u_i — in our case, the user who bought the product associated with the transaction $\vec{t_i}$. We represent the mapping of the properties of a transaction as functions of the form $f : \vec{t_i} \rightarrow u_i$. Importantly, we consider the mapping between the user identifier, u_i, and the transaction, $\vec{t_i}$, released in the output cluster, $\vec{t}_{cluster}$, as *sensitive*. Thus, we want to prevent any adversary $\mathcal{A}$ from learning this mapping or any other

linking relation of such a mapping to other transactions. There may exist other mapping functions such as the mapping of the price to a transaction that remains *non-sensitive* and may be revealed to an adversary or analyst to perform recommendations.

According to our system model, we consider β to be a released recommendation, which is calculated using the results $\vec{t}_{cluster}$ of the clustering step $\pi_{q,1}$ via a recommendation function, $\pi_{q,2}$[9]. Together, $\pi_{q,1}$ and $\pi_{q,2}$ form the vector of transformation function $\vec{\pi_q}$ of ϕ_{clu}. We require that both $\pi_{q,1}$ and $\pi_{q,2}$ are privacy-preserving.

Over the released clusters we can define the sets U_f, Q_f and P_f as follows. The participant set, U_f, contains all user identifiers, u_i, of all released transactions, $\vec{t_i} \in \vec{t}_{cluster}$. The usage frequency set, Q_f, contains a mapping between the user identifier, u_i, and the number of transactions, $\#u_i$, pertaining to that user, of all released transactions $\vec{t_i} \in \vec{t}_{cluster}$. Finally, the linking relation set, P_f, partitions all released transactions $\vec{t_i} \in \vec{t}_{cluster}$ regarding their associated user identifiers, u_i.

Analysis. The aim of a potential malicious analyst is to find or associate a user identifier to the transactions released in cluster $\vec{t}_{cluster}$. Considering the data to be released, Table 2 presents details of the privacy notions that can be achieved.

4.3. Collaborative-filtering based Classification

Consider a collaborative-filtering based recommendation system, ϕ_{col}. As discussed previously, ϕ_{col} is based upon the content-based clustering system, ϕ_{clu}, and takes as input a cluster of transactions, $\vec{t}_{cluster}$. We treat each single transaction $\vec{t_i} \in \vec{t}_{cluster}$ as a single input $\vec{\alpha_i}$. As the next step, the transformation functions $\vec{\pi_q}$ of ϕ_{col} execute the classification algorithm as described in Section 3.2 and release vectors of products, denoted by $\vec{e_i}$, for release query q_i and user u_i. Each row in this release pertains to one of the k-nearest neighbours of user u_i. Similar to the reasoning for ϕ_{clu}, the release products p_i still remain associated to their user identifier u_i via a function $f : p_i \rightarrow u_i$, though ϕ_{col} does not release any user identifier u_i in plain, but as a pseudonym, n_i. Pseudonym n_i may pertain to any user identifier u_i within the set of all user identifiers, $u_i \in U_i$. As such, any n_i is just the k-nearest neighbour mapping of a targeted user, u_j. We require that any potential malicious analyst $\mathcal{A}$ must not learn the mapping of $f : p_i \rightarrow u_i$ or the pseudonym mapping of $g : n_i \rightarrow u_i$, where f is the mapping function of outcome elements to user identifiers of our system model and g is a mapping function of pseudonyms n_i to the user identifiers u_i, where n_is are drawn from the same parameter space as u_is. We note that system ϕ_{col} only releases products p_is as outcome elements and that the associated pseudonyms n_i are presented here to provide a simplified understanding, due to the structure of the released products (*i.e.* p_is are ordered and reveal the linking relation P_f).

Again, according to our system model, we consider β to be a released recommendation. β is calculated using the resulting vector of products $\vec{e}$ of the k-nearest neighbours of a user u_i of the classification step $\pi_{q,1}$ via a recommendation function $\pi_{q,2}$. $\pi_{q,1}$ and $\pi_{q,2}$ form the vector of transformation function $\vec{\pi_q}$ of ϕ_{col}. We require that both $\pi_{q,1}$ and $\pi_{q,2}$ are privacy-preserving.

Over the released products of the k-nearest neighbours of user u_i, we can define the sets U_f, Q_f and P_f as follows. All user identifiers of the released products (or, indeed, all neighbour identifiers, $n_0, \ldots, n_k$) are included in the participant set, U_f. Similarly, the usage frequency set Q_f contains a mapping of all user identifiers u_i associated with the number of products $\#u_i$ bought by that user (and listed in the released products). Finally, the linking relation set P_f partitions all released products p_i regarding their associated user identifiers u_i[10].

Analysis. Again, the aim of a potential malicious analyst is to find or associate a user identifier to the products released of system ϕ_{col}. Considering the data to be released or inferable, Table 3 presents details of the privacy notions that can be achieved.

5. Analysis of Privacy Notions

In this section, we present the results of our theoretical analysis of the privacy games of [4]. We present limitations and probabilities on the likelihood of a release and an in/out adversary breaking any of the associated notions.

The games outlined in [4] provide the adversarial steps necessary to break an associated notion. However, it is not possible to predict which operations are executed by an adversary (nor their order). As such, the game-based definitions remain flexible: in detail, the games abstract only the absolute minimum steps necessary to break a notion. Any additional operations executed by the adversary are permitted, but not abstracted in a game. Thus, these games give an unambiguous understanding of the steps necessary to breach a privacy guarantee given by the associated notion.

In Figure 5 we outline procedures of game G. This is the basic game from which all other games are derived (all derived games can call the procedures of the basic game). Game G includes basic interfaces to system ϕ abstracted by procedures. For example, `initialise` resets all datasets in ϕ, `input` provides a user to input data values α_i and to specify which party p_i may store these values, `view` generates a release of ϕ and `corrupt` reveals the elements of a potentially corrupted party p_i. Further, within the games we use the notation $a \circ b$ and mean that a and b are combined (for example, via a concatenation operation). Yet, the games do not restrict the 'combination' function to the concatenation function — it may be any other (potentially, more complex) combination function. Also, we use the notation $a \xleftarrow{\$} b$ to denote a uniformly random assignment of a value from the parameter space b to the variable a.

9. In [4], we define $\vec{e}, \beta = \vec{\pi_q}(\vec{\alpha})$ (as such, indicating that β is generated over the inputs $\vec{\alpha}$). We want to clarify here, that we still comply with this convention. However, we note that, in [4], it is not specified how β is generated from $\pi_{q,2}$ and, as such, it may as well be generated using $\vec{t}_{cluster}$, which represents a transformed subset of $\vec{\alpha}$ (via function $\pi_{q,1}$).

10. In fact, due to the release mode of ϕ_{col}, P_f may always be identified, even if it is not accessible through interface $\mathcal{I}_{P_f}$, by simply observing the columns of ϕ_{col} released.

TABLE 2: Analysis of a content-based recommender system ϕ_{clu}.

Notion	✓/−	Comments
SA	✓	$\mathcal{A}$ only observes $\vec{t}_{cluster}$ and is given no additional information. By the requirement that f is not accessible, $\mathcal{A}$ does not gain any advantage to choose tuple (t_x, u_x) for interface $validate_{\text{SA}}$.
PH	✓	$\mathcal{A}$ observes $\vec{t}_{cluster}$ and is given access to $\lvert U_f \rvert$. $\mathcal{A}$ does not gain any advantage to select tuple (t_x, u_x) for interface $validate_{\text{PH}}$.
SU	✓	$\mathcal{A}$ observes $\vec{t}_{cluster}$ and is given access to U_f. $\mathcal{A}$ does not gain any advantage to select triple (t_x, t_y, u_x) for interface $validate_{\text{SU}}$.
WU	✓	$\mathcal{A}$ observes $\vec{t}_{cluster}$ and is given access to U_f and Q_f. $\mathcal{A}$ does not gain any advantage to select triple (t_x, t_y, u_x) for interface $validate_{\text{WU}}$.
PS	✓	$\mathcal{A}$ observes $\vec{t}_{cluster}$ and is given access to P_f. $\mathcal{A}$ does not gain any advantage to select tuple $(\vec{t_{u_x}}, u_x)$ for interface $validate_{\text{PS}}$.
AN	✓	$\mathcal{A}$ observes $\vec{t}_{cluster}$ and is given access to U_f and P_f. $\mathcal{A}$ does not gain any advantage to select tuple $(\vec{t_{u_x}}, u_x)$ for interface $validate_{\text{AN}}$.
WA	✓	$\mathcal{A}$ observes $\vec{t}_{cluster}$ and is given access to U_f, Q_f and P_f. $\mathcal{A}$ does not gain any advantage to select tuple $(\vec{t_{u_x}}, u_x)$ for interface $validate_{\text{WA}}$.

TABLE 3: Analysis of a collaborative-filtering based recommender system ϕ_{col}.

Notion	✓/−	Comments
SA	✓	$\mathcal{A}$ observes $\vec{e_i}$ and can infer P_f. Yet, $\mathcal{A}$ does not gain any advantage to select tuple (p_x, u_x) for interface $validate_{\text{SA}}$.
PH	✓	$\mathcal{A}$ observes $\vec{e_i}$, is given access to $\lvert U_f \rvert$, and can infer P_f. Yet, $\mathcal{A}$ does not gain any advantage to select tuple (p_x, u_x) for interface $validate_{\text{PH}}$.
SU	✓?	While $\mathcal{A}$ can easily pick p_x, p_y such that they map together, due to knowledge of P_f, she may not be able to map them to u_x. Thus, the notion may still be achieved, however, the probability for $\mathcal{A}$ to break it is higher.
WU	✓?	Same reasoning applies as for notion SU.
PS	✓	$\mathcal{A}$ observes $\vec{e_i}$ and is given access to (or, is able to infer) P_f. Yet, $\mathcal{A}$ does not gain any advantage to select tuple $(\vec{p_{u_x}}, u_x)$ for interface $validate_{\text{PS}}$.
AN	✓	$\mathcal{A}$ observes $\vec{e_i}$ and is given access to U_f and (and, is able to infer) P_f. Yet, $\mathcal{A}$ does not gain any advantage to select tuple $(\vec{p_{u_x}}, u_x)$ for interface $validate_{\text{AN}}$.
WA	✓	$\mathcal{A}$ observes $\vec{e_i}$ and is given access to U_f, Q_f and (and, is able to infer) P_f. Yet, $\mathcal{A}$ does not gain any advantage to select tuple $(\vec{p_{u_x}}, u_x)$ for interface $validate_{\text{WA}}$

proc$_{\mathcal{O}}$ $\mathcal{I}_{\mathbf{U_f}}$
  return U_f
proc$_{\mathcal{O}}$ $\mathcal{I}_{\mathbf{Q_f}}$
  return Q_f
proc$_{\mathcal{O}}$ $\mathcal{I}_{\mathbf{P_f}}$
  return P_f
proc initialise
  forall i in range$(0, n)$:
    $p_i.db \leftarrow \emptyset$

proc$_{\mathcal{O}}$ **input**(u_i, α_i, p_i)
  $p_i.db \leftarrow p_i.db \cup (u_i, \alpha_i)$
proc$_{\mathcal{O}}$ **view**$^{\pi_q}$
  forall i in range$(0, n)$:
    $(\vec{e}_i, \beta_i) \leftarrow \pi_q(p_i.db)$
  $\vec{e} \leftarrow \vec{e}_1 \circ \vec{e}_2 \circ \cdots \circ \vec{e}_n$
  $\beta \leftarrow \beta_1 \circ \beta_2 \circ \cdots \circ \beta_n$
  return $(\vec{e}, \beta)$
proc$_{\mathcal{O}}$ **corrupt**(p_i)
  return $p_i.db$

Figure 5: Game G (taken and adapted[11] from [4]).

Analysis (with regards to our case study). The application of differential privacy [1], [2] to recommender systems was previously introduced by McSherry and Mironov in [17]. Specifically, they consider the application and adaption of differential privacy to leading algorithms in the Netflix competition. However, most of the existing methods to preserve privacy based on differential private mechanisms protect user privacy at the cost of utility. In contrast, Yang *et al.* [6] adapt a privacy-preserving collaborative filtering method: they perturb the dataset and predict a user u_i's interest in products she has not rated yet. In detail, the perturbation must satisfy differential privacy (and, as such, prevent the k-nn attack), while ensuring that the result similarity of the perturbed and original dataset remains the same.

Consider a content-based clustering recommendation system, ϕ_{clu}. With the addition of differential private noise during the clustering process, it is possible to create 'privacy-enhanced' clusters that may be less predicable to an adversary observing the system. Yet, in order to satisfy data utility (in our context, to calculate a meaningful recommendation), it is necessary to maintain similarity of the clusters, which generated via a perturbed clustering algorithm[12] or a non-perturbed clustering algorithm.

Likewise, if we consider a collaborative-filtering based recommendation system. ϕ_{col}, then the addition of differential private noise during the classification process releases 'privacy-enhanced' nearest neighbours as discussed by Yang *et al.* [6], or by using 'privacy-enhanced' clusters as discussed in Section 4.3.

5.1. Generic Limitations of the Privacy Games

We have critically analysed the privacy notions of Bohli and Pashalidis [3] and their representation via games by the present authors [4] and identified the following limitations of the approach (with regards to adversary, system and privacy model; privacy notions; and privacy games) which may not be universally applicable to cover all privacy aspects of (privacy-preserving) systems. These generic limitations include:

1) **Release of multiple elements**: While the system model, in general, supports the release of only the aggregated value β, the privacy notions and our games are solely defined over the vector $\vec{e}$ (*i.e.* β is of no concern for the privacy notions/games).
2) **Released elements $e_i \in \vec{e}$ uniquely related to u_i**: In order to apply the privacy notions and games directly, each released element $e_i \in \vec{e}$ must be uniquely mapped to exactly one user identifier, u_i. There may be multiple elements in $\vec{e}$ which map to the same u_i, but there must not be an element $e_i \in \vec{e}$ which maps to two different user identifiers u_i, u_j, where $u_i \neq u_j$.

11. Game G here differs from game G of [4] in that the definition of the interfaces $\mathcal{I}_{U_f}, \mathcal{I}_{Q_f}$ and $\mathcal{I}_{P_f}$ is given in game G and not separately.

12. Under a 'perturbed' clustering algorithm, we understand that data collected for clustering is perturbed before the actual cluster generation process. As such, certain data points may or may not be included in the cluster, dependent on the level of noise addition.

3) **Privacy level dependent on type of (privacy-preserving) transformation function** $\pi_{q,i}$: The privacy games are defined to outline the necessary steps for an adversary to break an associated notion. However, the level of privacy achieved is dependent on the type of the transformation function $\pi_{q,i}$ (*e.g.* $\pi_{q,i}$ may be a secret permutation or perturbation based).
4) **Privacy model may not cover all privacy aspects**: While the privacy notions and games abstract a wide range of privacy issues, they may not cover all aspects of privacy (*e.g.* query privacy in the sense of differential privacy[13]).
5) **Adversary types and assumptions**: We consider an adversary who aims to learn the mapping function f to identify user identifiers from release elements e_i. There may be other privacy implications such as elements e_is may be sensitive itself. Moreover, in this paper we do not consider system adversaries, which may learn information about sensitive attributes from, for example, side channels[14]. Contrary, we assume that adversaries are semi-honest and follow the rules of the games. In real-world scenarios, it is not always possible to make such an assumption.
6) **Rigorous mathematical treatment and proofs**: While the system model, privacy notions and games follow a rigorous mathematical framework (to abstract a wide range of real-world systems), we do not formally prove systems secure (with respect to the games). This is partly due to missing definitions of 'idealistic' instances (of privacy-preserving systems) with regards to the privacy properties that are abstracted by our games (*e.g.* an idealistic instance with regards to unlinkability would always return unique 'unlinked' outcome values). Moreover, for example in our case study, we do not consider a concrete implementation of our privacy-preserving transformation function $\pi_{q,i}$ (*e.g.* $\pi_{q,i}$ is a perturbation function which adds random noise according to a given distribution). This is to remain abstract from concrete implementations to cover a wide range of applications (by abstracting the concrete mechanism and considering that data may be transformed; and, as such, be privacy-preserving).

5.2. Hierarchy, Limitations and Probabilities

In [3], Bohli and Pashalidis presented a hierarchy of their proposed privacy notions. We extend their hierarchy by the notions of unobservability, as illustrated in Figure 6.

In [4], the present authors illustrated relationships between privacy games by identifying groups of games with similar objectives. In detail, the privacy games may be categorised as follows: (SA, PH), (SU, WU), (PS, AN, WA) and (UO). These groups allow us to formulate informal 'reductions' between members of the same group. Concretely, we say if a system ϕ is secure under a 'weaker' notion within a group, it is also secure under a 'stronger' notion of the same group. For example, if ϕ is secure under PH it would imply that ϕ is also secure under SA. That is, because, for a weaker notion, an adversary $\mathcal{A}$ is given access to additional interfaces, while, for a stronger notion, $\mathcal{A}$ does not get this access. The learning goal of an adversary within a group remains the same. In our example, this would mean that for PH $\mathcal{A}$ would get additional access to the interface $\mathcal{I}_{|U_f|}$, while for SA she wouldn't. We leave formal proofs for such 'reductions' open for future work.

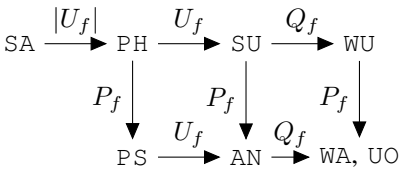

Figure 6: Hierarchy and relations between privacy notions. This graph extends the hierarchy given in [3] by the notion of unobservability. From left to right an adversary $\mathcal{A}$ is given access to interfaces and learns more information about f.

In the following analysis, we elaborate on the capabilities of adversaries and generic limitations of the games. In general, any adversary $\mathcal{A}$ may be limited in her ability to infer knowledge of a system. As such, she may be able to observe *any* interaction of a user with the system (*i.e.* input and release of data), *only* the released outcome, or certain aspects and properties of the systems release. The privacy notions and games proposed in [3] and [4] have been modelled to abstract such contexts within the release (by regulating access to sets U_f, Q_f and P_f).

We consider $\mathcal{A}$ to be of one of two classes: either of class $\mathcal{A}_{release}$ or of class $\mathcal{A}_{in/out}$. The former, a release adversary, is only able to access the released results of the system $\vec{e}, \beta$. The latter, an in/out adversary, is able to input data α_x into the system, observe other inputs $\vec{\alpha}$ to the system, as well as observe the released results of the system $\vec{e}, \beta$. Figures 7 and 8 visualise these kind of adversarial classes. We omit adversaries of the type, $\mathcal{A}_\phi$ (system adversaries), thus treating a system, ϕ (and its internal operations), as an assumable secure and privacy-preserving black box model. Such a kind of secure system may be achieved by trusted and hardware-enforced secure entities such as a SGX-based TRE [18]–[21] or any other strong cryptographically protected system.

In order to provide release privacy, any system ϕ must deploy a privacy-preserving transformation function $\vec{\pi_q}$. In our analysis, we assume that $\vec{\pi_q}$ is a privacy-preserving identity mapping. As such, it does not alter the input values $\vec{\alpha}$, but obfuscates the user identifier, u_i, from the released elements, $\vec{e}$. However, $\vec{\pi_q}$ may be a set of any kind of functions: for example, transforming multiple input values $\vec{\alpha}$ to a single outcome value, e_i. Thus, the probability of an adversary of any kind $\mathcal{A}_{release}$ or $\mathcal{A}_{in/out}$ to break any privacy notion needs to be assessed with this kind of

13. Differential privacy ensures that the addition of a element to a database (*i.e.* participation of an individual) doesn't change the query result significantly. In the context of the syntactic privacy games, a combination of the notions PH (participation hiding) and UO (unobservability) may closely resemble such privacy requirements.

14. Information gained from a side channel (attack) is by targeting a concrete implementation rather than any weakness in the algorithm.

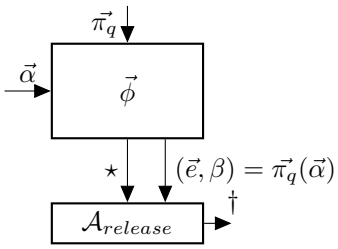

Figure 7: An adversarial model of type: release adversary[15]. This kind of adversary only obtains access to the system release $\vec{e}, \beta$.

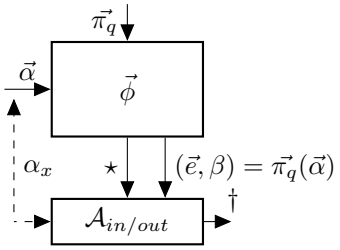

Figure 8: An adversarial model of type: in/out adversary[15]. This kind of adversary obtains access to the inputs $\vec{\alpha}$ as well as the system release $\vec{e}, \beta$.

transformation function $\vec{\pi_q}$ in mind.

We assume that $|\vec{e}| \geq |\vec{u}|$, with $|\vec{e}| = m$ and $|\vec{u}| = n$. Both m and n are 'large' integers — of an order such that an adversary, who has access to modern computational hardware should not be able to gain any advantage compared to randomly guessing[16]. Then, for a release adversary $\mathcal{A}_{release}$, under the assumption of a fixed user identifier u_x, the following probabilities can be observed:

(a) $Pr(\mathcal{A}^{\star}_{release} \Rightarrow \texttt{true}) = \frac{1}{|\vec{e}|-q}$, for games abstracting the notions $\star \in \{\texttt{SA}, \texttt{PH}\}$ with learning goal $f(e_x) = u_x$.

(b) $Pr[\mathcal{A}^{\star}_{release} \Rightarrow \texttt{true}] = \frac{1}{|\vec{e}|-q} * \frac{1}{|\vec{e}|-q}$, for games abstracting the notions $\star \in \{\texttt{SU}, \texttt{WU}\}$ with learning goal $f(e_x) = f(e_y) = u_x$ (assuming independence in the selection of e_x and e_y).

(c) $Pr[\mathcal{A}^{\star}_{release} \Rightarrow \texttt{true}] = \frac{|\vec{e_{u_x}}|}{|\vec{e}|-q}$, for games abstracting the notions $\star \in \{\texttt{PS}, \texttt{AN}, \texttt{WA}\}$ with learning goal $f(\vec{e_{u_x}}) = u_x$.

(For all games, q indicates the number of queries that $\mathcal{A}_{release}$ has submitted to the system ϕ. $|\vec{e}|$ denotes the size of the release elements — accounting the number of elements ϕ releases. Similarly, $|\vec{e_{u_x}}|$ accounts for the number of elements included in the release for a user u_x.)

For game $G_{\texttt{UO}}$ (unobservability), we would need to deploy a different measure to access the success probability to win said game. Concretely, such a measure could be the distinguishing advantage — though this requires the definition of an 'idealised' instance of the privacy-preserving system with regards to the aforementioned game. To assess the distinguishing advantage, it is evaluated if a 'real' instance is able to satisfy these guarantees (those outlined by the 'idealised' instance). To give an idea of such an approach, we roughly estimate $\mathcal{A}$'s probability to win game $G_{\texttt{UO}}$: we assume a release adversary ($\mathcal{A}$ is only give access to the outcome $(\vec{e}, \beta)$) and assume an 'idealistic' instance, where the target element $(\vec{e}_{u_x}, \beta_{u_x})$ is replaced by a random element from the same parameter space; the success probability is then estimated to be $Pr[\mathcal{A}^{\star}_{release} \Rightarrow \texttt{true}] = 0$, with $\star \in \{\texttt{UO}\}$. For a 'real' instance we assume that the generation of a replacement value would leak information (or may be detectable).

In the case of an in/out adversary, $\mathcal{A}_{in/out}$, the transformation functions $\vec{\pi}_q$ need to be strictly more privacy-preserving, since $\mathcal{A}_{in/out}$ has additional access to the inputs $\vec{\alpha}$ of a system, ϕ. Depending on the data within the system, $\mathcal{A}_{in/out}$ might submit any outlier elements α that are easily recognisable in the case of a privacy-preserving identity mapping. In such cases, $\vec{\pi}_q$ must transform the released elements (*e.g.* by a perturbation-based privacy mechanism). Otherwise, the same probabilities as in the case of a release adversary apply.

In some application scenarios and systems it is neither possible nor feasible to release elements with a large privacy margin. Therefore, to prevent an adversary from learning too much information, while still being able to maintain a smaller privacy margin and achieving certain privacy guarantees, it is possible to limit the access to any validate interfaces $\texttt{validate}_{\star}$. In this case, data must not be shared after a threshold of queries is reached in order to prevent loss of any individual's privacy.

6. Illustration of Privacy Properties

In Figure 9, we illustrate the privacy notions. As depicted, the system ϕ is queried with a triple (u_x, α_x, p_i) and releases a tuple $(\vec{e}, \beta)$ on a `view` query. Dashed encircled are the sets that the adversary $\mathcal{A}$ is given access to for each notion. Further, the dashed arrows show the objective of each notion. For simplicity, this representation includes only one database $p_i.db$. However, any finite number of databases are equally possible and supported by the model.

7. Conclusions

We set out to promote a better understanding of current syntactic privacy notions as proposed by Bohli and Pashalidis [3] and their adaption to privacy games by the present authors [4]. Since the presentation of [4] was largely theoretical, we demonstrated the applicability of the proposed privacy games to a case study based on content-based clustering and collaborative filtering-based classification. Via the case study we showed how the game-based privacy definitions have the potential to interconnect privacy implications and to reason about privacy

15. $\star$ indicates the knowledge $\mathcal{A}_{release}$ or $\mathcal{A}_{in/out}$ is given access to; further, † represents the learning goal of $\mathcal{A}_{release}$ or $\mathcal{A}_{in/out}$. These values change depending on the desired notion system ϕ is considered to achieve. A mapping of potential values associated to each privacy notion and game is given in the last two domains of Table 1.

16. If the release set $\vec{e}$ and the set of user identifiers $\vec{u}$ are sparsely populated an adversary may simply try any combination of variants to win a game. In this context, the elements pertaining to the sets $\vec{e}$, $\vec{u}$ should be selected from a range with cardinality comparable to modern cipher parameters (*e.g.* 2^{112} [22]). Under the assumption, that the adversary does not know any or only limited information about the contents of $\vec{e}$, $\vec{u}$, he/she should not gain any advantage compared to just randomly picking any element from the range.

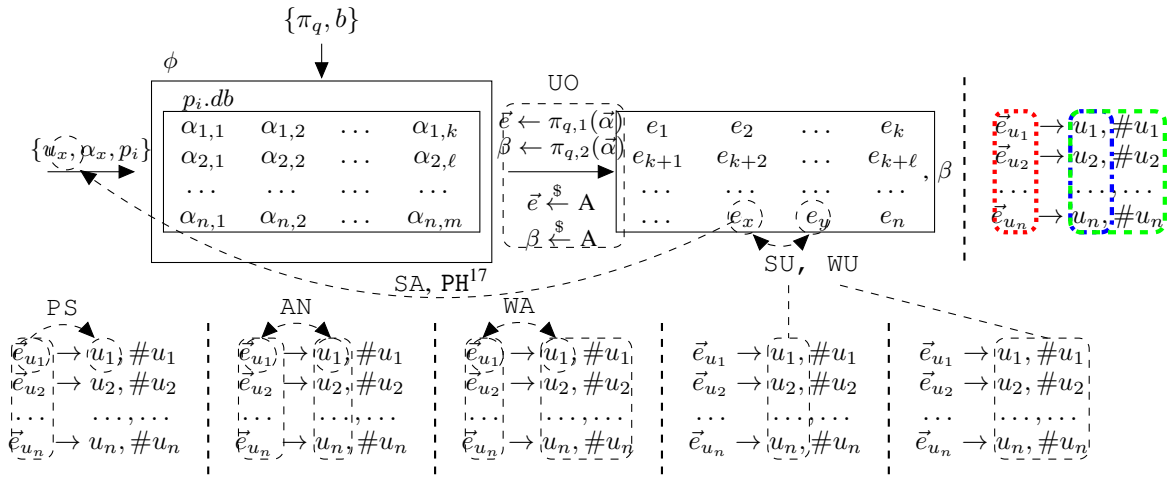

Figure 9: Illustration of privacy notions, given access to the system ϕ, and interface access to the sets $\mathbf{U_f}$ (▬▬), $\mathbf{Q_f}$ (▬ ▬) and $\mathbf{P_f}$ (▬▬).

properties within the context of recommender systems, as well as any other systems that may be representable using our system model.

This presentation was the first stake to apply our game-based privacy definitions of [4] to reason about privacy properties of a system. Future work may extend on this presentation to add a more detailed analysis and focus on systems in different environments and contexts.

We analysed the privacy games and presented limitations and probabilistic bounds for two classes of adversaries: release adversaries and in/out adversaries. Overall, we aim to provide a better understanding of privacy for non-experts such as system designers and software developers.

Additional avenues for future work include the extension of the privacy games and notions to fully distributed systems, policies for system designers to decide between certain notions and games, and the exploration of privacy-preserving transformation functions that guarantee the properties abstracted by the privacy notions.

References

[1] C. Dwork, "Differential privacy," in *ICALP '06*. Springer, 2006, pp. 1–12.

[2] ——, "Differential privacy: A survey of results," ser. TAMC '08. Springer, 2008, pp. 1–19.

[3] J.-M. Bohli and A. Pashalidis, "Relations among privacy notions," *ACM Trans. Inf. Syst. Secur.*, vol. 14, no. 1, pp. 4:1–4:24, 2011.

[4] R. Ankele and A. Simpson, "Abstracting syntactic privacy notions via privacy games," in *The 16th IEEE International Conference on Advanced and Trusted Computing 2019 (ATC 2019)*. Leicester, United Kingdom (Great Britain): IEEE, Aug. 2019.

[5] ——, "Analysis and evaluation of syntactic privacy notions and games," in *2018 16th Annual Conference on Privacy, Security and Trust (PST)*, Aug 2018, pp. 1–2.

[6] M. Yang, T. Zhu, L. Ma, Y. Xiang, and W. Zhou, "Privacy preserving collaborative filtering via the johnson-lindenstrauss transform," in *Trustcom '17*. IEEE, 2017, pp. 417–424.

[7] C. Ju and C. Xu, "A new collaborative recommendation approach based on users clustering using artificial bee colony algorithm," *The Scientific World Journal*, vol. 2013, p. 9, 2013.

17. For the notion of PH, $\mathcal{A}$ has additional access to an interface giving the size of the participation set $|\mathbf{U_f}|$.

[8] M. Bellare and P. Rogaway, "The game-playing technique," Cryptology ePrint Archive, Report 2004/331, 2004.

[9] D. Achenbach, M. Huber, J. Müller-Quade, and J. Rill, *Closing the Gap: A Universal Privacy Framework for Outsourced Data*. Cham: Springer International Publishing, 2016, pp. 134–151.

[10] Y. Lindell, "Anonymous authentication," *Journal of Privacy and Confidentiality*, vol. 2, no. 2, 2011.

[11] J. Alwen, M. Hirt, U. Maurer, A. Patra, and P. Raykov, "Anonymous authentication with shared secrets," in *Progress in Cryptology - LATINCRYPT 2014*, D. F. Aranha and A. Menezes, Eds. Cham: Springer International Publishing, 2015, pp. 219–236.

[12] K. Kluczniak, "Domain-specific pseudonymous signatures revisited," Cryptology ePrint Archive, Report 2016/070, 2016, https://eprint.iacr.org/2016/070.

[13] S. Ringers, E. Verheul, and J.-H. Hoepman, "An efficient self-blindable attribute-based credential scheme," Cryptology ePrint Archive, Report 2017/115, 2017, https://eprint.iacr.org/2017/115.

[14] J. A. Calandrino, A. Kilzer, A. Narayanan, E. W. Felten, and V. Shmatikov, ""you might also like:" privacy risks of collaborative filtering," in *SP '11*. IEEE, 2011, pp. 231–246.

[15] N. Ramakrishnan, B. J. Keller, B. J. Mirza, A. Y. Grama, and G. Karypis, "Privacy risks in recommender systems," *IEEE Internet Computing*, vol. 5, no. 6, pp. 54–63, 2001.

[16] B. Zhang, N. Wang, and H. Jin, "Privacy concerns in online recommender systems: Influences of control and user data input," ser. SOUPS '14. USENIX Association, 2014, pp. 159–173.

[17] F. McSherry and I. Mironov, "Differentially private recommender systems: Building privacy into the netflix prize contenders," ser. KDD '09. ACM, 2009, pp. 627–636.

[18] A. Paverd, "Enhancing Communication Privacy Using Trustworthy Remote Entities," D.Phil. dissertation, University of Oxford, 2016.

[19] R. Ankele, A. Küçük, A. Martin, A. Simpson, and A. Paverd, "Applying the trustworthy remote entity to privacy-preserving multiparty computation: Requirements and criteria for large-scale applications," in *ATC '16*. IEEE, 2016, pp. 414–422.

[20] A. Küçük, A. Paverd, A. Martin, N. Asokan, A. Simpson, and R. Ankele, "Exploring the Use of Intel SGX for Secure Many-Party Applications," ser. SysTEX '16. ACM, 2016, pp. 5:1–5:6.

[21] R. Ankele and A. Simpson, "On the performance of a trustworthy remote entity in comparison to secure multi-party computation," in *Trustcom '17*. IEEE, 2017, pp. 1115–1122.

[22] E. B. Barker and A. L. Roginsky, "Sp 800-131a. transitions: Recommendation for transitioning the use of cryptographic algorithms and key lengths," Tech. Rep., 2011.