Analysis and Evaluation of Syntactic Privacy Notions and Games

Robin Ankele and Andrew Simpson
Department of Computer Science, University of Oxford
Wolfson Building, Parks Road, OX1 3QD Oxford, United Kingdom

***Abstract*—Previous contributions have established a framework of privacy games that supports the representation of syntactic privacy notions such as anonymity, unlinkability, pseudonymity and unobservablility in the form of games. The intention is that, via such abstractions, the understanding of, and relationships between, privacy notions can be clarified. Further, an unambiguous understanding of adversarial actions is given. Yet, without any practical context, the potential benefits of these notions and games may be incomprehensible to system designers and software developers. We utilise these games in a case study based on recommender systems. Consequently, we show that the game-based definitions have the potential to interconnect privacy implications and can be utilised to reason about privacy.**

I. Introduction

In this paper, we utilise the privacy games presented by Ankele and Simpson [1] (which are based on the notions of Bohli and Pashalidis [2], which, in turn, are associated with privacy properties that a system can inherit). A drawback of the work of [2] is that these notions can be elusive, and prone to misunderstanding by system designers and software developers. The games presented in [1] enhance the understanding of, and characterise the relationships that hold between, these different notions. Yet, their presentation was largely theoretical: no case studies, use cases or examples were used to demonstrate applicability. We aim to bridge this gap.

We are particularly interested in two classes of recommender systems, based on *clustering* and *classification*. Clustering is the task of grouping sets of elements, while classification is the task of assigning new observations to a predefined set of classes. The case study was chosen because 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.

II. Background

A. System Model

A system ϕ takes inputs $\alpha_i \in \vec{\alpha}$, and releases outcome elements $e_i \in \vec{e}$ and a single (aggregated) value β. Each α_i and e_i is uniquely assigned to a user identifier u_i. Users interact with ϕ via interfaces — for example, an interface $\texttt{input}(\cdot,\cdot,\cdot)$ for inputs and $\texttt{view}^{\pi_q}(\cdot)$ to gain access to outputs. 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.

Within ϕ a set of transformation functions $\vec{\pi_q}$ map 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. This mapping may be non-injective, but we require that every outcome element must map to an input element. Transformation functions may also be composible: $\vec{\pi_q} = \pi_{q,1} \circ \pi_{q,2} \circ \cdots \circ \pi_{q,n}$.[1]

To reason about privacy, both [2] and [1] define a function $f : e_i \to u_i$, which associates a e_i with a u_i. This association is *sensitive* and should not be learned by an adversary $\mathcal{A}$. Using a privacy mechanisms may prevent such information gain. Furthermore, we inherit the definition of the sets U_f, Q_f and P_f from [2] and [1].

B. Privacy Notions and Privacy Games

We utilise the privacy notions of Bohli and Pashalidis [2]. Each privacy notion embodies its own restrictions or requirements on the released data of a system. We use the notions of strong anonymity ($\texttt{SA}$), participation hiding ($\texttt{PH}$), strong unlinkability ($\texttt{SU}$), weak unlinkability ($\texttt{WU}$), pseudonymity ($\texttt{PS}$), anonymity ($\texttt{AN}$), weak anonymity ($\texttt{WA}$) and unobservability ($\texttt{UO}$). The reader is referred to [2] for a precise definition of each notion.

In [1] each privacy notion is associated with a game that an adversary must win in order to break the privacy notion. $\mathcal{A}$ gains access to a limited number of interfaces $\mathcal{I}$ to obtain knowledge as well as the system output $(\vec{e}, \beta)$. An adversary, $\mathcal{A}$, 'breaks' a privacy notion if she is able to come up with a 'strategy' 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 associated with the 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 $\star \in \{\texttt{SA}, \texttt{PH}, \texttt{SU}, \texttt{WU}, \texttt{PS}, \texttt{AN}, \texttt{WA}, \texttt{UO}\}$. The reader is referred to [1] for a more detailed description of the privacy games.

III. A Case Study: Recommender Systems

A. Content-based Clustering

Consider a system, ϕ_{clu}, which recommends products based on previous transactions, as illustrated in Figure 1. Here, ϕ_{clu}

[1]We define $(\vec{e}, \beta) = \vec{\pi_q}(\vec{\alpha})$ and say 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 $(\vec{e}, \beta)$.

978-1-5386-7493-2/18/$31.00 ©2018 IEEE

TABLE I
ANALYSIS OF RECOMMENDER SYSTEMS USING THE PRIVACY NOTIONS AND GAMES OF [2] AND [1].

Notion	(ϕ_{clu}/ϕ_{col})	ϕ_{clu}	ϕ_{col}
SA	✓/ ✓	$\mathcal{A}$ only observes $\vec{t}_{cluster}$ and is given no additional information. f is not accessible, thus $\mathcal{A}$ does not gain any advantage to choose tuple (t_x, u_x) for $validate_{\texttt{SA}}$.	$\mathcal{A}$ 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 $validate_{\texttt{SA}}$.
PH	✓/ ✓	$\mathcal{A}$ observes $\vec{t}_{cluster}$ and is given access to $\|U_f\|$. $\mathcal{A}$ does not gain any advantage to select tuple (t_x, u_x) for $validate_{\texttt{PH}}$.	$\mathcal{A}$ observes $\vec{e_i}$, is given access to $\|U_f\|$, and can infer P_f. Yet, $\mathcal{A}$ does not gain any advantage to select tuple (p_x, u_x) for $validate_{\texttt{PH}}$.
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 $validate_{\texttt{SU}}$.	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	✓/ $✓_?$	$\mathcal{A}$ 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 $validate_{\texttt{WU}}$.	Same reasoning applies as for notion SU.
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 $validate_{\texttt{PS}}$.	$\mathcal{A}$ 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 $validate_{\texttt{PS}}$.
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 $validate_{\texttt{AN}}$.	$\mathcal{A}$ 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 $validate_{\texttt{AN}}$.
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 $validate_{\texttt{WA}}$.	$\mathcal{A}$ 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 $validate_{\texttt{WA}}$

takes transactions $\vec{t_i}$ as input values $\vec{\alpha_i}$. Next, ϕ_{clu} performs a set of transformation functions $\vec{\pi_q}$ to release a vector of transactions: cluster $\vec{t}_{cluster}$ (equivalent to $\vec{e_i}$), for query q_i.

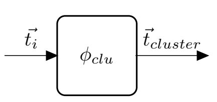
β is the released recommendation, calculated over the results $\vec{t}_{cluster}$. Over the released clusters we can define the sets U_f, Q_f and P_f as follows. The participant set, U_f, contains all user identifiers, u_i, of all released transactions, $\vec{t_i} \in \vec{t}_{cluster}$. The usage frequency 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 I presents details of the privacy notions that can be achieved.

B. Collaborative-filtering based Classification

Consider a collaborative-filtering based recommendation system, ϕ_{col}, as illustrated in Figure 2. Here, ϕ_{col} is based upon the content-based clustering system, ϕ_{clu}, and takes as input a cluster of transactions, $\vec{t}_{cluster}$. These clusters are used 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. For the recommendation, we use a simple majority function over the k-nearest neighbours to recommend products to the users.

Again, we treat transaction $\vec{t_i} \in \vec{t}_{cluster}$ as inputs $\vec{\alpha_i}$. Here, ϕ_{col} executes a classification algorithm and releases a vectors of products (equivalent to $\vec{e_i}$). Each row in this release pertains to one of the k-nearest neighbours of user u_i, but is 'hidden' via a pseudonym n_i. We require that any potential malicious analyst $\mathcal{A}$ must not learn the mapping of $f : p_i \to u_i$ or the pseudonym mapping of $g : n_i \to u_i$, 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 the values of n_i are drawn from the same parameter space as the values of u_i.



$$\vec{t}_{cluster} = \begin{Bmatrix} \vec{t_1} & \vec{t_2} & \dots & \vec{t_k} \\ \vec{t}_{k+1} & \vec{t}_{k+2} & \dots & \vec{t}_{k+\ell} \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \vec{t_m} \end{Bmatrix}$$

Fig. 1. 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 $\vec{t_i}.product$.

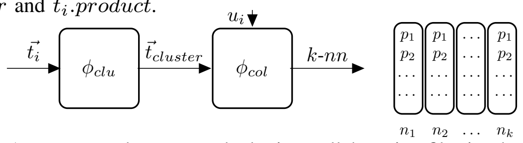


Fig. 2. A recommender system deploying collaborative-filtering based classification: in the first step transactions $\vec{t_i}$ are clusterised into clusters $\vec{t}_{cluster}$; in the second step, for a user u_i, the products p_i of all k-nearest neighbours $(n_1, \dots, n_k)$ are released. Recommendations can be generated via a majority vote over the products p_i among all neighbouring users.

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. All user identifiers of the released products (or, indeed, all neighbour identifiers, $n_0, \dots, n_k$) 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.[2]

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 I presents details of the privacy notions that can be achieved.

REFERENCES

[1] R. Ankele and A. Simpson, "Privacy games for syntactic privacy notions," Cryptology ePrint Archive, Report 2017/1126, 2017.

[2] 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.

[2]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.