Abstracting Syntactic Privacy Notions via Privacy Games

Robin Ankele and Andrew Simpson
Department of Computer Science, University of Oxford
Wolfson Building, Parks Road, Oxford OX1 3QD, United Kingdom
robin.ankele@cs.ox.ac.uk, andrew.simpson@cs.ox.ac.uk

Abstract—**It is well understood that the huge volumes of data captured in recent years have the potential to underpin significant research developments in many fields. But, to realise these benefits, all relevant parties must be comfortable with how this data is shared. At the heart of this is the notion of privacy — which is recognised as being somewhat difficult to define. Previous authors have shown how privacy notions such as anonymity, unlinkability and pseudonymity might be combined into a single formal framework. We use and extend this work by defining privacy games for individual and group privacy within distributed environments. For each privacy notion, we formulate a game that an adversary has to win in order to break the notion. Via these games, we aim to clarify understanding of, and relationships between, different privacy notions; we also aim to give an unambiguous understanding of adversarial actions. Additionally, we extend previous work via the notion of *unobservability*.**

Index Terms—**Privacy Notions, Privacy Games, Release Privacy, Unobservability**

1. Introduction

The continuous digitalisation of our daily lives has led to the collection of huge volumes of data. The sharing and use of that data can be beneficial in many ways. However, challenges arise from the processing of such data, one of which is guaranteeing privacy to individuals and groups. One of the complexities in this respect is defining privacy, which, by its very nature, can mean different things to different people — resulting in a variety of notions capturing different perspectives on and properties of privacy.

Early efforts at establishing a consistent terminology in this regard include those of Pfitzmann and Köhntopp [1], who developed a precise terminology for privacy notions. Despite such efforts, an open problem with the formal treatment of privacy notions remained: discrepancies between formal models led to ill-defined relationships and incomparable notions. In addition, various attacks [2]–[6] influenced and challenged the formulation of new privacy notions. Bohli and Pashalidis addressed those disconnected notions in [7], as a first step to unite multiple privacy notions into a single formal framework.

We build upon [7] by giving consideration to the challenges faced in distributed, heterogeneous contexts. For each privacy notion, we formulate a game an adversary has to win in order to break the notion. Via these games, we aim to clarify understanding of, and relationships between, different privacy notions; we also aim to give an *unambiguous understanding* of adversarial actions. We also extend the privacy framework of [7] by introducing the notion of *unobservability*. Concretely, this notion guarantees to prevent an adversary from distinguishing if a system invocation has taken place or not (by observing a real or 'generated' outcome element). As such, this notion gives similar guarantees as the prominent notion of differential privacy [8]. However, instead of dealing with probabilistic privacy notions, we focus on syntactical notions for privacy. These notions are associated to the privacy properties that a system can inherit: they are defined on properties of the system's outcome (*i.e.* data to be released by the system).

The presented notions and games may be used to reason about privacy properties in privacy-preserving systems. As such, we consider systems that may be stateful, stateless, or online. Moreover, we focus on individual and group privacy for interactive, non-interactive and statistical release scenarios. Nevertheless, while the presented notions and games can address a wide range of privacy abstractions, not all issues may be covered by our definitions. As such, some restrictions are inherent from our system model and our adversary model.

The kind of system of interest to us is manifested by the deployment of Trustworthy Remote Entities (TREs) [9], [10]. A TRE provides input privacy by leveraging novel hardware security extensions, such as Intel's SGX[1] [11], [12] extensions, and may deploy differential privacy mechanisms to achieve release privacy.

2. Syntactic Privacy vs. Probabilistic Privacy

Defining privacy and notions representing specific properties related to privacy must be undertaken with care. Some privacy definitions ignore the knowledge of an adversary, and only give a requirement of the structure of the resulting data. These notions pertain to the class of *syntactic* methods, which are mainly concerned with the syntactic form of the released outcome. In our context, ignorance of a priori knowledge of an adversary, often called *background knowledge* or *auxiliary information*, may easily lead to privacy breaches if the adversary is able to obtain and combine information from other sources.

Many privacy attacks[2] focus on the model of non-interactive privacy, due to its release-and-forget approach

1. Intel Software Guard Extensions (SGX), https://software.intel.com/en-us/sgx

2. Attacks targeting non-interactive mechanisms include linking attacks (for example, homogeneity attacks [2], skewness attacks [3] or similarity attacks [3]), composition attacks [4] and minimality attacks [5].

(*i.e.* 're-sanitisation' of a released (sanitised) database in case of a privacy breach is not possible). Mechanisms to provide privacy in this setting have evolved, but this has subsequently led to the problem of these algorithms becoming increasingly complex. Examples for non-interactive release privacy include [13] and [3].

Another class of privacy notions pertains to *probabilistic* methods. Contrary to syntactic methods, these notions capture the probabilities that govern the relationship between inputs to outputs. For example, the notion of *differential privacy* (DP) [8] helps provide privacy in the interactive setting. DP allows for privacy-preserving views over statistical databases, while providing provable guarantees that the distribution of noisy query answers changes only negligibly with addition or deletion of any tuple in the database. However, while the mathematics of differential privacy are straightforward, an intuitive understanding of the notion can be elusive, as can characterisations of its relationships with, for example, anonymity, participation hiding or unlinkability. Therefore, many variations have been proposed to provide different assurances[3]

Frameworks that aim to unify privacy notions (mainly focused on probabilistic privacy) include Pufferfish [15] and the framework of membership privacy [16]. Importantly, definitions within these frameworks satisfy the fundamental axioms for modern privacy design guidelines: *transformation invariance* and *convexity* [17]. Domingo-Ferrer [18] proposed the notion of co-privacy, a game-theoretic approach, with the attractive property that the best strategy of a player is to help preserving the privacy of other players. A similar approach was defined by Sánchez *et al.* [19], who proposed the notion of co-utility. In co-utile protocols, a player increases her utility by helping others to increase theirs. Privacy games defined in these frameworks aim to identify the 'best' strategy for their players to receive the best outcome. A player's incentive is to follow this strategy, thus enhancing privacy/utility of the overall protocol outcome. Alternatively, the games defined in our context are aimed to provide non-experts with a better understanding of the underlying privacy notions. Within the context of syntactic privacy notions, to provide a clear understanding of privacy, its properties and implications to systems, it is important to accurately model adversarial actions. Game-based notions formulate games illustrating the necessary actions for an adversary to break said notion. Such game-based definitions have been widely used in the cryptography community to provide simpler and more easily verifiable proofs. Examples include [20], [21]. Moreover, such games are also widely used to argue about privacy properties within anonymous authentication protocols. Examples include games abstracting notions of unforgeability, seclusiveness, anonymity or unlinkability in pseudonymous signatures schemes [22] and attribute-based credential schemes [23]. Similarly, via our games, we aim to clarify understanding of, and relationships between, different privacy notions. In this context, in [24] we present an analysis of the privacy games, limitations of the approach and, importantly, a first stake to apply the presented game-based definitions in a case study based on recommender systems.

3. Contributions that present criticisms and relaxations of the pure mode of DP are summarised in [14].

3. The Framework of Bohli and Pashalidis

Many privacy models assume that data to be shared originates from a centralised database. Thus, users must always trust the data curator to keep their data secure (and, consequently, preserve their privacy). Such models, however, do not capture use cases associated with distributed environments, where data is split over several devices and is stored in databases *distributed* over a large network.

3.1. Privacy Model

While their system model is clearly focused on a centralised system, Bohli and Pashalidis [7] do not explicitly state any assumptions pertaining to the underlying (centralised or distributed) structure of their privacy model. Indeed, the privacy definitions follow the syntactic approach: as such, certain privacy notions may be achieved dependent on the information an adversary gains access to (or is allowed to learn). They do, though, consider systems that follow the non-interactive release mode. Batches of input invocations trigger a sequence of outputs, where output values represent a permutation of the input values. In particular, this permutation obfuscates the association to a user identifier — essentially, the goal, in order to release data in a privacy-preserving way. Furthermore, [7] gives considerations to stateful, stateless and online systems.

In their system model, Bohli and Pashalidis model a function that maps the serial number of the output values to the identifier of the user associated with it. By means of this function, they define privacy properties that abstract the relation between the output element and the user identifier. Moreover, in [7] Bohli and Pashalidis show how these notions are related to each other.

3.2. Privacy Notions

The privacy notions considered in this paper are formally defined in Section 6, but we introduce them first in terms of natural language to aid comprehension. (Table 1 provides an overview of the privacy notions in terms of our system model and states the learning goals of an adversary $\mathcal{A}$ and knowledge of $\mathcal{A}$.) Informally, we say that an adversary (simplified) is allowed to interact[4] with a system as follows:

(a) it inputs information, and/or
(b) it obtains an outcome — in our case, a vector of input values (permuted, to ensure privacy) and an aggregated value (computed over the input values).

Then, the objectives of the privacy notions introduced in [7] are as follows:

· *Strong anonymity* prevents an adversary from learning any interesting relationship, from the observation of the outcome, with regards to the input.
· *Participation hiding* prevents an adversary from learning the relation of an outcome value to a specific user.
· *Unlinkability* prevents an adversary from learning (*i.e.* identifying) any outcome elements that are related to each other (*i.e.* pertaining to the same user).

4. This is not a one-shot interaction, but a user (or adversary) may access the system interfaces multiple times.

- *Pseudonymity* allows an adversary to associate groups of outcome elements with a pseudonym, but prevents her from learning the actual user identifier.
- *Anonymity* prevents an adversary from learning of the relationship between groups of elements and/or groups of user identifiers.

Additionally, the objective of our extended notion of unobservability is as follows:

- *Unobservability* prevents an adversary from distinguishing if a system invocation has taken place or not (by observing a real or 'generated' outcome element).

3.3. An Extension to a Semi-Distributed Model

A fully-distributed system represents a collection of single systems. As such, data is to be stored at each of the distributed parties and the privacy properties are manifested in each system's individual release and in the combined release. A user interacts via separate `input` and `view` interfaces of each individual system. Consequently, the benefits of such an approach are: (a) data remains at the data provider; (b) a reduced communication overhead; and (c) no single point of failure. Examples of such (privacy-preserving) systems include usage patterns collections (*e.g.* the approach by Apple[5] in iOS 10).

Yet, reasoning about privacy properties in such systems may be complex and challenging. One needs to take into consideration that an adversary may obtain, corrupt or control a certain threshold of system outcomes. Furthermore, privacy notions need to be abstracted to include such 'collaborative' outcomes and relations between outcomes of individual systems. Thus, as a first step, we consider a semi-distributed model. Data is still split over multiple parties, however users gain access to the system via a global `input` and `view` interface. As such, privacy properties may be considered only over the whole 'combined' system release. In other words, we consider the underlying structure as a 'black box'. Yet, we consider data to be distributed among multiple systems (as such, that the games to abstract the privacy notions take such a distributed modelling into consideration).

3.3.1. Individual and Group Privacy. An adversary may be able to infer knowledge about an individual by directly or by indirectly targeting her (*e.g.* by grouping users according to similar properties such as their behaviour, location or user type; and gaining knowledge from the observation of the group). The former case we further denote as breaching an *individual's privacy* and the latter as breaching *group privacy*.

3.3.2. Stateful, Stateless and Online Systems. The design and application of privacy notions depends heavily on the underlying structure of a system, be it *stateful*, *stateless*, or *online*. We discuss examples following a permutation as privacy-preserving mechanism — yet any other mechanisms may apply similarly. In a stateful system, released data may be shuffled according to a state- (or data-) dependent, potentially secret, permutation. Additionally, statistical queries may result in state-dependent answers (for example, identical queries may not be answered, result in the same answer, or be dependent on the privacy mechanism to include an increased noise value). In a stateless system, release data may be shuffled according to a permutation chosen uniformly at random. Because stateless systems may not rely on any state (from the perspective of the system), queries submitted to it are independent and any input has no influence on the behaviour of the system. In the design of privacy notions and privacy-preserving mechanisms we must cope with such restrictions (for example, submission of two identical queries might disclose information). Finally, in an online system, data is processed and output individually (*i.e.* before an invocation of another input), due to its limited buffer size. Therefore, a permutation may not be applicable and other sources (such as noise addition) may need to be applied to preserve privacy in such a setting.

5. https://wired.com/2016/06/apples-differential-privacy-collecting-data

4. System Model

We consider systems, denoted by ϕ, that are sequentially invoked a finite number of times, and require that they return values $e_i \in \vec{e}$ (where $\vec{e}$ denotes the set of all released elements) and an aggregated value β computed over a finite number of inputs $\alpha_i \in \vec{\alpha}$ (where $\vec{\alpha}$ denotes the set of all input elements). We also require that each input α_i must be associated to a unique user identifier u_i and a computational party p_i. Inputs α_i associated to the same user identifier u_i may collectively be denoted in vector notation $\vec{\alpha}_{u_i}$. The same applies for the output values e_i, with $\vec{e}_{u_i}$ for outputs associated to the same u_i. Tuples of the form (u_i, α_i) are stored within datasets at the specified party p_i. The inputs to, as well as the outputs from, ϕ are drawn from a parameter space $\alpha_i, e_i \in A$ that is system-specific. Similarly defined is the parameter space of the aggregated output $\beta \in A$. Each user identifier u_i and party identifier p_i is assumed to be unique, and drawn from an identifier space.

Invocations of ϕ, in the form of a query q_i, produce output batches $(\vec{e}_i, \beta_i)$ of potentially varying size. Thus, a query q_i is essentially a mapping of input batches to $(\vec{e}_i, \beta_i)$, denoted by π^{ϕ} for values $\vec{e}_i$ and π_{q_i} for the aggregated value β_i. In particular, ϕ accepts input batches of the form $(u_1, \alpha_1, p_1), (u_2, \alpha_2, p_2), \dots, (u_c, \alpha_c, p_c)$ for c invocations of the input interface $\texttt{input}(\cdot, \cdot, \cdot)$. In order to preserve a user u_i's privacy, ϕ applies for each invocation of the output interface $\texttt{view}^{\pi_\cdot}(\cdot)$, for example, a secret permutation π^{ϕ} on the datasets held by each party p_i and outputs a sequence $(e_1, \dots, e_c), \beta$ where $\vec{e} \leftarrow \pi^{\phi}(\vec{\alpha})$ and $\beta \leftarrow \pi_{q_i}(\vec{\alpha})$. We do not limit π^{ϕ} to be a secret permutation, but π^{ϕ} may be any privacy-preserving mapping function.

In the above, we assumed that each input (u_i, α_i) is distributed to a different party p_i. However, we do not limit our model in such contexts, but allow a data owner[6] to freely decide to which party p_i she wants to

6. Data submitted to the system may include (or pertain to) several data subjects, that are independent of the actual data owner. We assume that each input value α_i is associated to a user identifier u_i, which identifies the data subject (which may or may not be the data owner). We further assume that data subjects have given consent to data owners to perform certain data-mining tasks on the committed data, while preserving the privacy of the data subjects. In any case, the data owner decides with which party p_i to share the data.

TABLE 1: Summary of the notions for privacy considered in our privacy model: strong anonymity (SA), participation hiding (PH), strong unlinkability (SU), weak unlinkability (WU), pseudonymity (PS), anonymity (AN), weak anonymity (WA), and unobservability (UO). The notions are grouped regarding similar objectives in distinct categories: single, multiple and element groups, as well as system behaviour. Further, for each category we indicated the learning goal of an adversary. In this context, the knowledge of $\mathcal{A}$ indicates access to certain interfaces of the system ϕ depending on the privacy notion under question. (Further details are given in Section 6.)

Privacy Notion	Category	Learning Goal	Knowledge of $\mathcal{A}$
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		$f(\vec{e}_{u_x}) = u_x$	P_f
AN	element groups	$f(\vec{e}_{u_x}) = u_x$	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

send her inputs. Our model is assumed to handle any data distribution among parties p_i.

Building on the model of [7], our model includes the possibility of a 'void' invocation of the system, *i.e.* an invocation of ϕ, where the outcome $(\vec{e}, \beta)$ is unrelated to any u_i, α_i or p_i and drawn uniformly at random from the parameter space A. However, we require that the outcome of a void invocation must remain indistinguishable to the outcome of a 'normal' invocation of ϕ. This property is essential in the context of system unobservability. Our model allows single users or a collection of users to reason about their privacy assurances. Precisely, a notion or game can be applied to capture privacy properties about a mapping between:

1) a single user and an attribute: $u_x \rightarrow e_{u_x}$
2) a single user and multiple attributes: $u_x \rightarrow (e_{u_x,1}, \dots, e_{u_x,c})$
3) multiple users and multiple attributes: $u_x \rightarrow e_{u_x}, u_y \rightarrow (e_{u_y,1}, \dots, e_{u_y,c})$

where the last case denotes group privacy.

We define the functions f and $f^\star$ as follows. $f : \vec{e} \rightarrow u_i$ takes as an input an element (or vector) e_i (or $\vec{e}$) of ϕ and maps it to the corresponding user u_i. Similarly, $f^\star : \vec{\alpha}_i \rightarrow u_i$, with α_i (or $\vec{\alpha}_i$) (stored in the dataset of any party p_i). Through the processing of a query q_i (*i.e.* essentially through π_{q_i}), ϕ obfuscates the function f. An adversary $\mathcal{A}$ must not be able to directly learn f by observing the inputs (u_i, α_i, p_i) to ϕ or the released outcome $(\vec{e}, \beta)$. Nevertheless, the adversary's goal remains to determine f, or any other interesting property about f. Still, the privacy notions and games reveal, to varying degrees, information about f.

We denote $p_i \in P$, with P the set of all computational parties. Inputs of ϕ are stored in datasets $\alpha_i \rightarrow p_i.db$, given any possible distribution. Moreover, we say $\vec{\alpha} = \cup_{i=1}^{P} p_i.db$, that is, $\vec{\alpha}$ is the set of all inputs $\alpha_i \in \vec{\alpha}$ for all parties p_i. Similarly, it holds that $\vec{e} = \cup_{i=1}^{n} e_i$ represents the union of all outcome values. Following from the definitions of [7], we denote the following properties of the outcome in the form of sets:

Definition 1. *(Participant Set):* $U_f = \{f(e_i) : e_i \in \vec{e}\}$ denotes the set of participants in ϕ. Each $e_i \in \vec{e}$ is uniquely associated to 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 $e_i \in \vec{e}$ each user u_i is associated with. In other words, $\#u_i$ is the number of e_i 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 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 $\forall e_i \in \vec{e}_{u_i} : f(e_i) = u_i$.

Remark 1. For a fixed user u_x, we denote the subsets $U_{f_x} \subseteq U_f, Q_{f_x} \subseteq Q_f$ and $P_{f_x} \subseteq P_f$, where $U_{f_x} = u_x, Q_{f_x} = (u_x, \#u_x)$ and $P_{f_x} = \vec{e}_{u_x} = (e_{x,1}, .., e_{x,k})$. We assume that $\mathcal{A}$, by knowledge of U_{f_x}, Q_{f_x} and P_{f_x}, in combination, can *uniquely* determine (or evaluate) the mapping function $f(e_x) = u_x$ and therefore *uniquely* map $e_x \rightarrow u_x$. However, this holds only in this limited setting, and with the assumption that $\mathcal{A}$ obtains the above sets. By knowledge of the generalised (system-wide) parameters U_f, Q_f and P_f, $\mathcal{A}$ may not be able to uniquely map $f(e_x) = u_x$, due to the fact that equivalence classes in Q_f do not have to be unique. For example, if the input distribution is uniform (*i.e.* each user contributes the same number of input values to the system), $\mathcal{A}$ would not gain any knowledge about the mapping of $f(e_x) = u_x$ by solely observing U_f, Q_f and P_f.

5. Adversarial Model and Principles of Privacy Games

5.1. Adversarial Model

An adversary, denoted by $\mathcal{A}$, is represented by an algorithm that runs in polynomial time. $\mathcal{A}$ is given access to a limited number of interfaces, as defined by the privacy game. An interface, denoted by $\mathcal{I}$, gives $\mathcal{A}$ access to knowledge about ϕ and relations between the outcome values $(\vec{e}, \beta)$. The interaction of $\mathcal{A}$ with an interface $\mathcal{I}$ is further denoted by $\mathcal{A}^{\mathcal{I}}$.

We say that $\mathcal{A}$ is interested in learning the mapping function f, or any interesting properties about f, represented by the defined notions in Section 6. Each of these notions is represented as a game. We define a game G, and say that $\mathcal{A}$ is able to break the privacy notion associated with that game if G evaluates

```
$\mathbf{proc}_{\circlearrowleft}\ \mathcal{I}_{\mathbf{U_f}}$
    forall $i$ in range$(0, |\vec{e}|)$:
        $U_f \leftarrow U_f \cup f(e_i)$
    return $U_f$
```

Figure 1: Interface $\mathcal{I}_{U_f}$ (Participant Set).

```
$\mathbf{proc}_{\circlearrowleft}\ \mathcal{I}_{\mathbf{Q_f}}$
    forall $i$ in range$(0, |\vec{e}|)$:
        $Q_f \leftarrow Q_f \cup \{f(e_i)\}$
    return $U_f$
```

Figure 2: Interface $\mathcal{I}_{Q_f}$ (Usage Frequency Set).

```
$\mathbf{proc}_{\circlearrowleft}\ \mathcal{I}_{\mathbf{P_f}}$
    $P_f \leftarrow$ part$(\vec{e})$
    return $P_f$
```

Figure 3: Interface $\mathcal{I}_{P_f}$ (Linking Relation).

```
$\mathbf{proc}_{\circlearrowleft}\ \mathcal{I}_{|\mathbf{U_f}|}$
    $|U_f| \leftarrow$ sizeof$(\mathcal{A}^{\mathcal{I}_{U_f}})$
    return $|U_f|$
```

Figure 4: Interface $\mathcal{I}_{|U_f|}$ (Number of Participants).

to `true` with a non-negligible probability. The phrasing 'non-negligible' is here defined with respect to the size of the anonymity set (*i.e.* number of users / elements within the system ϕ). As such, we require $|\vec{e}|$ and $|U_f|$ to be large[7], as otherwise $\mathcal{A}$ would have always a non-negligible success probability. The success probability of $\mathcal{A}$ is denoted by $\mathbf{Succ}(\mathcal{A}) = Pr[G_\star^\mathcal{A} \rightarrow \texttt{true}]$, where $\star \in \{\texttt{SA}, \texttt{PH}, \texttt{SU}, \texttt{WU}, \texttt{PS}, \texttt{AN}, \texttt{WA}, \texttt{UO}\}$[8] denotes any of the games. Another measure, for example, the distinguishing advantage of $\mathcal{A}$ would require the definition of 'idealised' instances of privacy-preserving systems with regards to the aforementioned properties (*e.g.* an idealised instance with regards to unlinkability would always return unique 'unlinked' outcome elements).

We assume $\mathcal{A}$ to be semi-honest: when $\mathcal{A}$ 'plays' a game, we assume that it behaves truthfully and follows the game's rules. In other words, $\mathcal{A}$ does not deviate from the game. In particular, this means that $\mathcal{A}$ does not submit elements that it has not previously obtained from the system ϕ (*e.g.* it does submit a random value $e_x \notin \vec{e}$, where $\vec{e} \leftarrow \mathcal{A}^{\texttt{view}^{\pi_q}()}$), or any of the other interfaces $\mathcal{I}_{U_f}, \mathcal{I}_{Q_f}, \mathcal{I}_{P_f}$ or $\mathcal{I}_{|U_f|}$, revealing specific information about subsets of f.

Precisely, $\mathcal{A}$ is allowed to interact adaptively with the interfaces of ϕ, and, at a given point in time, must commit an input value pair $(u_x, \vec{\alpha}_x)$ to which it receives the output tuple $(\vec{e}_x, \beta_x)$ from ϕ, which we further denote as *general observation*. Given these parameters (and access to some system-specific interfaces, as defined by the notions below), it will be evaluated if $\mathcal{A}$ is able to successfully break the given notion. For individual privacy, we assume u_x to be fixed; for group privacy, we assume $u_x \in \vec{u}_g$, where $\vec{u}_g$ is a fixed group of user identifiers.

These restrictions allow us to abstract (and in some cases omit) some of ϕ's specific operations and tasks in order to keep the following games concise and clearly readable. To this end, we do not include checks of validity (for example, $u_x \in U_f$, for input or validate procedures) for any of the values submitted to the interfaces. The interfaces $\mathcal{I}_{U_f}, \mathcal{I}_{Q_f}, \mathcal{I}_{P_f}$ and $\mathcal{I}_{|U_f|}$ are shown in Figures 1–4

5.2. Principles of Privacy Games

A game $G_\star$ consists of a limited number of procedures. A procedure, denoted by $\mathbf{proc}$, defines the computational steps an algorithm has to follow in order to terminate. Procedures are executed top-down and in order. Further, some procedures may only be executed once during an attack session, and others may be executed an arbitrary number of times, indicated by $\mathbf{proc}_{\circlearrowleft}$. This allows $\mathcal{A}$ to 'play around a bit' and get used to the system, before, for a fixed set of inputs, it tries to break the privacy notion, by executing $\mathbf{proc}\ \star$, where $\star \in \{\texttt{SA}, \texttt{PH}, \texttt{SU}, \texttt{WU}, \texttt{PS}, \texttt{AN}, \texttt{WA}, \texttt{UO}\}$.

```
$\mathbf{proc\ initialise}$
    forall $i$ in range$(0, n)$:
$p_i.db \leftarrow \emptyset$
$\mathbf{proc}_{\circlearrowleft}\ \mathbf{corrupt}(p_i)$
    return $p_i.db$
$\mathbf{proc}_{\circlearrowleft}\ \mathbf{view}^{\pi_q}$
    forall $i$ in range$(0, n)$: $\vec{e}_i \leftarrow \pi^\phi(p_i.db)$; $\beta_i \leftarrow \pi_q(pi.db)$
    $\vec{e} \leftarrow \vec{e}_1 \circ \vec{e}_2 \circ \cdots \circ \vec{e}_n$
    $\beta \leftarrow \beta_1 \circ \beta_2 \circ \cdots \circ \beta_n$
    return $(\vec{e}, \beta)$

$\mathbf{proc}_{\circlearrowleft}\ \mathbf{input}(u_i, \alpha_i, p_i)$
    $p_i.db \leftarrow p_i.db \cup$
$(u_i, \alpha_i)$
```

Figure 5: Game G.

We define a basic game G, which provides procedures used by all games $G_\star$. We define *inheritance* as follows: if game G_x inherits all procedures of game G_y, denoted $G_x \vDash G_y$, then G_x can, additionally to its own procedures, call any of the procedures of G_y. In our case: $G_\star \vDash G$. In Figure 5, we present game G. $\mathcal{A}$ may play game G for an *arbitrary* number of times, before it plays a derived game $G_\star$ *once*, where she aims to break the expressed notion.

In general, $\mathcal{A}$ interacts with ϕ ('plays' the games) in the following way:

1) $\mathcal{A}$ inputs data (α_i) into the system ϕ.
2) $\mathcal{A}$ obtains[9] output $(\vec{e}, \beta)$ from the system ϕ.
3) $\mathcal{A}$ may gain additional information by corrupting a threshold of parties.
4) $\mathcal{A}$ selects any value from the output, to which it believes it knows the relation (specified by the privacy notions) and submits that to the validate interface (*e.g.* any $e_x \in \vec{e}_x$, where $\mathcal{A}$ believes to know u_x).
5) Game $G_\star$ returns `true` or `false` according to the output of the validate interface.

6. Privacy Games

We now state the notions for privacy that we consider in our privacy model. For each notion, we formulate a game that an adversary has to win in order to break the associated notion. The privacy notions follow and extend the work of Bohli and Pashalidis [7]; the game-based definitions are novel. In formulating said notions and games, we take into account that they must abstract the behaviour of a system (*i.e.* it may be stateful, stateless or online), must abstract the locality of the computational parties (*i.e.*

7. *i.e.* it would be computationally infeasible to perform random guesses.

8. 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). See [7] for precise definitions of each notion.

9. $\mathcal{A}$ obtains $(\vec{e}, \beta)$ by querying the `view` interface. Moreover, $\vec{e}_x \circ \vec{e}_y$, for example, denotes the composition of two sets $\vec{e}_x$ and $\vec{e}_y$ under function π^ϕ. This means that $\pi^\phi(\vec{e}_x) \circ \pi^\phi(\vec{e}_y)$ is equivalent to $\pi^\phi(\vec{e}_x \cup \vec{e}_y)$. Similarly, for β, $\circ$ denotes the π_q operation.

```
proc validate$_{SA}(e_x, u_x)$
    if $f(e_x) = u_x$ then: return true else: return false
proc SA
    $\mathcal{A}^{\mathtt{input}(u_x,\alpha_x,p_x)}$
    $\vec{e}_x, \beta_x \leftarrow \mathcal{A}^{\mathtt{view}^{\pi_q}()}$
    return $\mathcal{A}^{\mathtt{validate_{SA}}(e_x,u_x)}$
```

Figure 6: Game $G_{\mathtt{SA}}$ (Strong Anonymity).

```
proc$_{\circlearrowleft}$ $\mathcal{I}_{|\mathbf{U_f}|}$
    return $|U_f|$
proc validate$_{PH}(e_x, u_x)$
    if $f(e_x) = u_x$ then: return true else: return false
proc PH
    $\mathcal{A}^{\mathtt{input}(u_x,\alpha_x,p_x)}$
    $\vec{e}_x, \beta_x \leftarrow \mathcal{A}^{\mathtt{view}^{\pi_q}()}$
    return $\mathcal{A}^{\mathtt{validate_{PH}}(e_x,u_x)}$
```

Figure 7: Game $G_{\mathtt{PH}}$ (Participation Hiding).

centralised or distributed), and must give consideration to individuals, as well as groups.

In Figures 6-13, we illustrate the games $G_\star$, with $\star \in \{\mathtt{SA}, \mathtt{PH}, \mathtt{SU}, \mathtt{WU}, \mathtt{PS}, \mathtt{AN}, \mathtt{WA}, \mathtt{UO}\}$. Most of the games have similarities (*e.g.* differ only in the validate interface or differ by the access to interfaces $\mathcal{I}_{U_f}, \mathcal{I}_{Q_f}, \mathcal{I}_{P_f}$ or $\mathcal{I}_{|U_f|}$). Thus, we group games and notions thus: (SA, PH), (SU, WU), (PS, AN, WA) and (UO).

6.1. Privacy Games for Single Elements

Figures 6 and 7 show $G_{\mathtt{SA}}$ (strong anonymity; see Def. 4) and $G_{\mathtt{PH}}$ (participation hiding; see Def. 5) respectively. The objective of group (SA, PH) is, from the outcome $\vec{e}_x, \beta_x$, to find any value $e_x \in \vec{e}_x$ for which $\mathcal{A}$ is able to identify the user identifier u_x. The validate functions of this group return true if $\mathcal{A}$ is able to do so.

Definition 4 (Strong Anonymity [7])**.** A system is said to provide strong anonymity (SA) if it does not leak any information about the mapping function f.

Definition 5 (Participation Hiding [7])**.** A system is said to hide participation (PH) if it does not leak any information about f beyond the size of the participant set $|U_f|$.

A system ϕ that satisfies the property of (SA, PH) prevents $\mathcal{A}$ from relating output elements to their associated user identifiers. More precisely, $\mathcal{A}$ must not be able, for any $e_x \in \vec{e}_x$, to learn the associated identifier u_x, such that $f(e_x) = u_x$. In the case of SA, $\mathcal{A}$ is not given anything beyond the general observation. In the case of PH, in addition to the general observation, we assume that $\mathcal{A}$ is able to learn $|U_f|$.

6.2. Privacy Games for Multiple Elements

Figures 8 and 9 show $G_{\mathtt{SU}}$ (strong unlinkability; see Def. 6) and $G_{\mathtt{WU}}$ (weak unlinkability; see Def. 7) respectively. The objective of group (SU, WU) is, from the outcome $\vec{e}_x, \beta_x$, to find any two values $e_x, e_y \in \vec{e}_x$ for which $\mathcal{A}$ believes they are related to the user identifier u_x. The validate functions of this group return true if $\mathcal{A}$ is able to do so.

```
proc$_{\circlearrowleft}$ $\mathcal{I}_{\mathbf{U_f}}$
    return $U_f$
proc validate$_{SU}(e_x, e_y, u_x)$
    if $f(e_x) = f(e_y) = u_x$ then: return true else: return
false
proc SU
    $\mathcal{A}^{\mathtt{input}(u_x,\alpha_x,p_x)}$
    $\mathcal{A}^{\mathtt{input}(u_x,\alpha_y,p_x)}$
    $\vec{e}_x, \beta_x \leftarrow \mathcal{A}^{\mathtt{view}^{\pi_q}()}$
    return $\mathcal{A}^{\mathtt{validate_{SU}}(e_x,e_y,u_x)}$
```

Figure 8: Game $G_{\mathtt{SU}}$ (Strong Unlinkability).

```
proc$_{\circlearrowleft}$ $\mathcal{I}_{\mathbf{U_f}}$           proc$_{\circlearrowleft}$ $\mathcal{I}_{\mathbf{Q_f}}$
    return $U_f$                 return $Q_f$
proc validate$_{WU}(e_x, e_y, u_x)$
    if $f(e_x) = f(e_y) = u_x$ then: return true else: return
false
proc WU
    $\mathcal{A}^{\mathtt{input}(u_x,\alpha_x,p_x)}$
    $\mathcal{A}^{\mathtt{input}(u_x,\alpha_y,p_x)}$
    $\vec{e}_x, \beta_x \leftarrow \mathcal{A}^{\mathtt{view}^{\pi_q}()}$
    return $\mathcal{A}^{\mathtt{validate_{WU}}(e_x,e_y,u_x)}$
```

Figure 9: Game $G_{\mathtt{WU}}$ (Weak Unlinkability).

Definition 6 (Strong Unlinkability [7])**.** A system is said to provide strong unlinkability (SU) if it does not leak any information about f beyond the participant set U_f.

Definition 7 (Weak Unlinkability [7])**.** A system is said to provide weak unlinkability (WU) if it does not leak any information about f beyond the usage frequency set Q_f[10].

A system ϕ that satisfies the property of (SU, WU) prevents $\mathcal{A}$ from relating output elements that pertain to the same user identifier. More precisely, $\mathcal{A}$ must not be able, for a fixed user identifier u_x to find any two elements $e_x, e_y \in \vec{e}_x$, such that $f(e_x) = f(e_y) = u_x$. In the case of SU, in addition to the general observation, we assume that $\mathcal{A}$ is able to learn U_f. In the case of WU, in addition to the general observation, we assume that $\mathcal{A}$ is able to learn U_f and Q_f. This allows $\mathcal{A}$ to learn, for each user, the number of elements of its outcome set.

6.3. Privacy Games for Element Groups

Figures 10–12 show $G_{\mathtt{PS}}$ (pseudonmyity; see Def. 8), $G_{\mathtt{AN}}$ (anonymity; see Def. 9) and $G_{\mathtt{WA}}$ (weak anonymity; see Def. 10) respectively. The objective of group (PS, AN, WA) is, from the outcome $\vec{e}_x, \beta_x$, given the ability to group outcome elements together, for this very group $\vec{e}_{u_x}$, to identify the user identifier u_x. The validate functions of this group return true if $\mathcal{A}$ is able to do so.

Definition 8 (Pseudonymity [7])**.** A system is said to provide pseudonymity (PS) if it does not leak any information about f beyond the linking relation P_f.

Definition 9 (Anonymity [7])**.** A system is said to provide anonymity (AN) if it does not leak any information about f beyond the participation set U_f and the linking relation P_f.

Definition 10 (Weak Anonymity [7])**.** A system is said to provide weak anonymity (WA) if it does not leak any

10. Note that knowledge of Q_f implicitly implies knowledge of U_f.

information about f beyond the participation set U_f, the usage frequency set Q_f and the linking relation P_f.

A system ϕ that satisfies the property of (PS, AN, WA) prevents $\mathcal{A}$ to relate a group of output elements (or an element within that group), which pertain to the same user identifier, to the user identifier. More precisely, $\mathcal{A}$ must not be able, for a fixed user identifier u_x, to find the group of related outcome elements $\vec{e}_{u_x} \subseteq \vec{e}_x$ such that $\forall e_x \in \vec{e}_{u_x} | f(e_x) = u_x$. In the case of PS, in addition to the general observation, we assume that $\mathcal{A}$ is able to learn P_f. This allows $\mathcal{A}$ to assign a pseudonym to each partition of the outcome set $\vec{e}_x$. In the case of AN, in addition to the general observation, we assume that $\mathcal{A}$ is able to learn U_f and P_f. This allows $\mathcal{A}$ to learn the user identifiers associated with the outcome values, and the ability to learn partitions of the overall outcome set $\vec{e}_x$, but not the relationship between those sets. Finally, in the case of WA, in addition to the general observation, we assume that $\mathcal{A}$ is able to learn U_f, Q_f and P_f[11]. This allows $\mathcal{A}$ to learn the user identifiers associated with the outcome values, partitions of the overall outcome set $\vec{e}_x$ and information about the number of elements each user has contributed in the outcome set.

6.4. Unobservability or Privacy Games for System Behaviour

Finally, Figure 13 shows $G_{\mathtt{UO}}$ (unobservability; see Def. 11). The objective of group (UO) is, from the outcome $\vec{e}_x, \beta_x$, to distinguish if the values $\vec{e}_{u_x}, \beta_{u_x}$ were generated as a result of a system invocation, or generated purely at random. The validate functions of this group return `true`, if $\mathcal{A}$ is able to do so.

Definition 11 (Unobservability)**.** A system is said to be unobservable (UO) if the output is indistinguishable from a random output in the range of the output parameter set, even with knowledge of the participation set U_f, the usage frequency set Q_f and the linking relation P_f.

A system ϕ that satisfies the property of UO prevents $\mathcal{A}$ to distinguish that, for a fixed tuple (u_x, α_x) of user identifier and input value, a system invocation has taken place or not, *i.e.* $\mathcal{A}$ is able to observe the overall system outcome $\vec{e}_x$, where all elements $\vec{e}_x \backslash \{e_x\}$ are result of a system invocation of $\vec{e}_x \leftarrow \pi^{\phi}(\vec{\alpha}_x)$ except for the fixed input value α_x. In other words, with system invocation we understand that ϕ outputs a single element e_x. In this context, it may be easy to distinguish a system ϕ behaving in the above way, by simply determining $|\vec{e}_x|$. If $\alpha_x \notin \vec{\alpha}_x$, then $|\vec{e}_x|_{\alpha_x \notin \vec{\alpha}_x} < |\vec{e}_x|_{\alpha_x \in \vec{\alpha}_x}$, where in the latter case $\alpha_x \in \vec{\alpha}_x$. Therefore, even for a *void* invocation of ϕ, it must hold $\exists e_x \in \vec{e}_x | e_x \not\leftarrow \pi^{\phi}(\alpha_x)$. The value e_x must be produced in a way that is indistinguishable of a system being invoked or not (in our game-based definition we say the outcome element gets assigned to a random value from the parameter space, *i.e.* $e_x \xleftarrow{\$} A$). Consequently, to achieve unobservability, $\mathcal{A}$ must not be able to distinguish if $\vec{e}_x \leftarrow \pi^{\phi}(\vec{\alpha}_x), \beta_x \leftarrow \pi_q(\vec{\alpha}_x)$ given $\alpha_x \in \vec{\alpha}_x, f^{\star}(\alpha_x) = u_x$, or, given $\alpha_x \notin \vec{\alpha}_x, f^{\star}(\alpha_x) \neq u_x, e_x \xleftarrow{\$} A$.

In addition to the general observation, we assume that $\mathcal{A}$ is able to learn U_f, Q_f and P_f. This allows $\mathcal{A}$ to learn the user identifiers associated with the outcome values, partitions of the overall outcome set $\vec{e}_x$ and information about the number of elements each user has contributed in the outcome set.

Remark 2. For group privacy, we assume $u_x \in \vec{u}_g$, where $\vec{u}_g$ is a group of fixed user identifiers. Then, each objective

11. In the case that each user contributes a unique amount of elements to ϕ, and $\mathcal{A}$ is able to infer this knowledge, then it is easy to see that $\mathcal{A}$ is able to determine $f(e_x) = u_x$ by the mere combination of the sets U_f, Q_f and P_f. However, in other cases, where this condition is not satisfied (for example, the number of elements each user contributes is uniformly distributed), $\mathcal{A}$ only partially gains information by the combination of the previously mentioned sets.

```
proc_○ $\mathcal{I}_{\mathbf{P_f}}$
    return $P_f$
```

```
proc validate_PS($\vec{e}_{u_x}, u_x$)
    if $f(\vec{e}_{u_x}) = u_x$ then: return true else: return false
proc PS
    $\mathcal{A}^{\mathtt{input}(u_x,\alpha_x,p_x)}$
    $\vec{e}_x, \beta_x \leftarrow \mathcal{A}^{\mathtt{view}^{\pi_q}}()$
    return $\mathcal{A}^{\mathtt{validate_{PS}}(\vec{e}_{u_x},u_x)}$
```

Figure 10: Game $G_{\mathtt{PS}}$ (Pseudonymity).

```
proc_○ $\mathcal{I}_{\mathbf{U_f}}$              proc_○ $\mathcal{I}_{\mathbf{P_f}}$
    return $U_f$                       return $P_f$

proc validate_AN($\vec{e}_{u_x}, u_x$)
    if $f(\vec{e}_{u_x}) = u_x$ then: return true else: return false
proc AN
    $\mathcal{A}^{\mathtt{input}(u_x,\alpha_x,p_x)}$
    $\vec{e}_x, \beta_x \leftarrow \mathcal{A}^{\mathtt{view}^{\pi_q}}()$
    return $\mathcal{A}^{\mathtt{validate_{AN}}(\vec{e}_{u_x},u_x)}$
```

Figure 11: Game $G_{\mathtt{AN}}$ (Anonymity).

```
proc_○ $\mathcal{I}_{\mathbf{U_f}}$              proc_○ $\mathcal{I}_{\mathbf{Q_f}}$
    return $U_f$                       return $Q_f$
proc_○ $\mathcal{I}_{\mathbf{P_f}}$              proc validate_WA($\vec{e}_{u_x}, u_x$)
    return $P_f$                       if $f(\vec{e}_{u_x}) = u_x$ then:
                                   return true else: return
                                   false
proc WA
    $\mathcal{A}^{\mathtt{input}(u_x,\alpha_x,p_x)}$
    $\vec{e}_x, \beta_x \leftarrow \mathcal{A}^{\mathtt{view}^{\pi_q}}()$
    return $\mathcal{A}^{\mathtt{validate_{WA}}(\vec{e}_{u_x},u_x)}$
```

Figure 12: Game $G_{\mathtt{WA}}$ (Weak Anonymity).

```
proc_○ $\mathcal{I}_{\mathbf{U_f}}$              proc_○ $\mathcal{I}_{\mathbf{Q_f}}$
    return $U_f$                       return $Q_f$
proc_○ $\mathcal{I}_{\mathbf{P_f}}$              proc validate_UO($g$)
    return $P_f$                       return $(g == b)$
proc_○ view$^{\pi_q}_{\mathbf{refut.}}$
    $b \xleftarrow{\$} \{0,1\}$
    forall $i$ in range$(0,n)$: $\vec{e}_i \leftarrow \pi^{\phi}(p_i.db)$; $\beta_i \leftarrow \pi_q(pi.db)$
    if $b = 0$ then: $\vec{e}_{u_x} \xleftarrow{\$} A$; $\beta_{u_x} \xleftarrow{\$} A$
    $\vec{e} \leftarrow \vec{e}_1 \circ \vec{e}_2 \circ \cdots \circ \vec{e}_n$
    $\beta \leftarrow \beta_1 \circ \beta_2 \circ \cdots \circ \beta_n$
    return $(\vec{e}, \beta)$
proc UO
    $\mathcal{A}^{\mathtt{input}(u_x,\alpha_x,p_x)}$
    $\vec{e}_x, \beta_x \leftarrow \mathcal{A}^{\mathtt{view}^{\pi_q}_{\mathbf{refut.}}}()$
    return $\mathcal{A}^{\mathtt{validate_{UO}}(g)}$
```

Figure 13: Game $G_{\mathtt{UO}}$ (Unobservability).

may be extended to find elements $e_x \in \vec{e}_x$ that relate to any u_x pertaining to the group of $\vec{u}_g$. For example, if $\mathcal{A}$ was about to infer if any e_x is related with e_y, where $f(e_x) = u_x$, $f(e_y) = u_y$ and $u_x, u_y \in \vec{u}_g$, then this would violate group privacy.

6.5. Relationships between Privacy Games

Table 1 provides an overview of the privacy notions and states the learning goals of an adversary $\mathcal{A}$; in particular, the privacy notions are listed with respect to a partial order dependent on the knowledge of $\mathcal{A}$.

In each game we allow an adversary $\mathcal{A}$ to gain access to interfaces. From this access and knowledge of the outcome values $(\vec{e}, \beta)$, $\mathcal{A}$ infers knowledge about the function f. The games are presented in order such that each subsequent game obtains more information about f (via access to multiple interfaces). In [7], Bohli and Pashalidis outline this hierarchy, which also applies to our privacy games (our notion of unobservability can be joined in this hierarchy with the notion of weak anonymity).

Our grouping of the privacy games ((SA, PH), (SU, WU), (PS, AN, WA) and (UO)) helps to identify games with similar objectives. In addition, if a system is secure under a 'weaker' notion within each of these groups is also secure under a 'stronger' notion in the same group, since the weaker notions provide access to additional interfaces while for the stronger notion $\mathcal{A}$ doesn't have access to it. For example, if ϕ is secure under PH this implies that it also secure under SA, since, for SA, $\mathcal{A}$ doesn't have access to $\mathcal{I}_{|U_f|}$.

7. Conclusions

We have built upon the syntactic notions of other authors and formulate those notions into privacy games, following the game-playing technique of provable security. We extended the framework of [7] by the notion of unobservability. With the definition of these games, we aim to clarify understanding of, and relationships between, different privacy notions; we also aim to give an unambiguous understanding of adversarial actions (in order to win a game and, therefore, break a notion). Our goal with the abstraction as privacy games is to provide a 'different' way of looking at privacy notions. The visual representation is aimed especially at non-experts such as software developers and software architects — to foster their understanding of privacy and get a better grip of the complex environments. Thus, we also take into consideration the different requirements of privacy protections for individuals and groups. A first take to apply the game-based definitions is presented in [24] for a case study on recommender systems. Many existing models for privacy focus on the preservation of privacy in centralised environments, though various use cases show the importance for concise privacy models and notions in distributed environments. Our notions, games and system model are designed for distributed environments (yet are still compliant with centralised systems). Future work will include an extension to externally distributed systems (reasoning about privacy aspects between the release of various systems), policies for the selection of privacy notions, exploration of primitives guaranteeing such notions and, importantly, application to real-world case studies.

References

[1] A. Pfitzmann and M. Köhntopp, "Anonymity, unobservability, and pseudonymity — a proposal for terminology," in *Designing PETs*. Springer, 2001, pp. 1–9.

[2] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, "L-diversity: Privacy Beyond K-anonymity," *ACM Trans. Knowl. Discov. Data*, 2007.

[3] N. Li, T. Li, and S. Venkatasubramanian, "t-Closeness: Privacy Beyond k-Anonymity and l-Diversity," in *ICDE '07*. IEEE, 2007.

[4] S. R. Ganta, S. P. Kasiviswanathan, and A. Smith, "Composition attacks and auxiliary information in data privacy," ser. KDD '08. ACM, 2008, pp. 265–273.

[5] R. C.-W. Wong, A. W.-C. Fu, K. Wang, and J. Pei, "Minimality attack in privacy preserving data publishing," ser. VLDB '07. VLDB, 2007, pp. 543–554.

[6] D. Kifer, "Attacks on privacy and definetti's theorem," ser. SIGMOD '09. ACM, 2009, pp. 127–138.

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

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

[9] 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.

[10] 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.

[11] I. Anati, S. Gueron, S. Johnson, and V. Scarlata, "Innovative Technology for CPU based Attestation and Sealing," ser. HASP '13, vol. 13. ACM, 2013.

[12] F. McKeen, I. Alexandrovich, A. Berenzon, C. V. Rozas, H. Shafi, V. Shanbhogue, and U. R. Savagaonkar, "Innovative Instructions and Software Model for Isolated Execution," ser. HASP '13, 2013.

[13] L. Sweeney, "K-anonymity: A Model for Protecting Privacy," *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 2002.

[14] P. Cuff and L. Yu, "Differential privacy as a mutual information constraint," ser. CCS '16. ACM, 2016, pp. 43–54.

[15] D. Kifer and A. Machanavajjhala, "Pufferfish: A framework for mathematical privacy definitions," *ACM Trans. Database Syst.*, vol. 39, no. 1, pp. 3:1–3:36, 2014.

[16] N. Li, W. Qardaji, D. Su, Y. Wu, and W. Yang, "Membership privacy: A unifying framework for privacy definitions," ser. CCS '13. ACM, 2013, pp. 889–900.

[17] D. Kifer and B.-R. Lin, "Towards an axiomatization of statistical privacy and utility," ser. PODS '10. ACM, 2010, pp. 147–158.

[18] J. Domingo-Ferrer, "Coprivacy: Towards a theory of sustainable privacy," in *Proceedings of the 2010 International Conference on Privacy in Statistical Databases*, ser. PSD'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 258–268.

[19] D. Sánchez, J. Domingo-Ferrer, and S. Martnez, "Co-utile disclosure of private data in social networks," *Inf. Sci.*, vol. 441, no. C, pp. 50–65, May 2018.

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

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

[22] K. Kluczniak, "Domain-specific pseudonymous signatures revisited," Cryptology ePrint Archive, Report 2016/070, 2016.

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

[24] R. Ankele and A. Simpson, "Analysing and evaluating syntactic privacy games via a recommender systems case study," in *The 16th IEEE International Conference on Advanced and Trusted Computing 2019 (ATC 2019)*. IEEE, Aug. 2019.