Privacy Notions for Privacy-Preserving Distributed Data Mining:
Foundations and Privacy Games

Robin Ankele and Andrew Simpson

University of Oxford, Department of Computer Science, Wolfson Building, Parks Road, OX1 3QD Oxford, United Kingdom
{robin.ankele, 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. In this paper we use and extend this work by defining privacy games for individual and group privacy within distributed environments. More precisely, 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*.

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. By its very nature, privacy is multi-dimensional and multi-faceted, and 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 colleagues [28,27], 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. Bohli *et al.* addressed those disconnected notions in [7], as a first step towards uniting multiple privacy notions into a single formal framework.

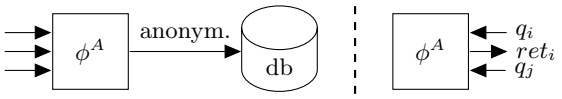

Fig. 1: Illustration of the two privacy-preserving data release modes: non-interactive (left) and interactive (right).

We build upon the work of Bohli *et al.* [7] and give consideration to the challenges faced in distributed, heterogeneous contexts. In particular, we extend the privacy framework of Bohli *et al.* by introducing the notion of *unobservability* and give consideration to individual and group privacy for interactive, non-interactive and statistical release scenarios — therefore reflecting a wide scope of applications. In addition, we consider systems that may be stateful, stateless, or online. 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.

These privacy notions and games may be used in privacy-preserving systems to reason about their privacy properties. The kind of system of interest to us is manifested by the deployment of Trustworthy Remote Entities (TREs) [4,18,26] — which we use to motivate our contribution. A TRE is able to provide input privacy by leveraging novel hardware security extensions, such as Intel's SGX[1] [3,23,8] extensions and, further, deploy differential privacy mechanisms to achieve release privacy.

Overall, we claim the following contributions. We present games illustrating notions for individual and group privacy in distributed settings. Moreover, we extend the privacy model of Bohli *et al.* [7] by the notion of *unobservability*.

2 Motivation and Background

2.1 Privacy-Preserving Data Release

In general, we consider two natural models in the privacy-preserving release of data: *non-interactive* and *interactive*, as illustrated in Figure 1. In the non-interactive setting, we assume data is to be shared by a trusted data curator, who publishes a 'safe' or 'sanitised' version of the database. In the literature this process is also know as *de-identification* or *anonymisation* of data. Traditionally, this can be achieved by the deployment of generalisation, suppression or perturbation mechanisms. Non-interactive data release gained in popularity in the mid 2000s with developments such as k-anonymity [29], ℓ-diversity [22] and t-closeness [19]. In the interactive setting, we assume users submit queries to the data curator. The data curator applies the query function on the database to obtain the so-called *true* answer. To protect privacy, for example, random noise is added to the true answer, with the subsequent result being returned to the user. Although there exist many other mechanisms to preserve privacy,

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

the utilisation of data perturbation seems both natural and appealing (and predominantly used for explicit notions [12]). This can come in two flavours: input perturbation and output perturbation. While the former adds noise to the data before submission to the data curator, the latter adds noise only after the query mapping is applied (and added to the true answer). The first approach is stricter (in the sense of preserving privacy, although it limits utility) and also works if the data curator is untrusted. The latter requires a trustworthy data curator, but typically achieves more utility. Statistical data release is a variant — or, in fact, a specialisation — of the interactive approach: the query function computes and returns a single value containing statistical information about the database entries. The notion of differential privacy [10] emerged as a means of ensuring privacy in such contexts.

2.2 A Motivating Example: Trustworthy Remote Entities

A Trustworthy Remote Entity (TRE) [26] is a system that enables privacy-preserving data release. The concept was introduced by Paverd [26] and extended by Ankele *et al.* [4] and Küçük *et al.* [18] to the application of privacy-preserving data release in distributed settings. In its simplest form, a TRE operates as an intermediary between participating communication parties. Essentially a TRE portrays a trusted third party (TTP), but, contrary to a TTP that requires users to blindly trust all performed actions, a TRE can verify its trustworthy nature. This can be done by using Intel Software Guard Extensions (SGX) [3,23,8] as an underlying hardware security primitive, which features isolated execution of code and memory pages in so-called *enclaves*, encrypted storage outside of the immediate CPU package, and software attestation to promote its trustworthiness. A TRE is not bound to any architectural design choice and may be applied in a centralised or distributed fashion. Overall, a TRE provides input privacy to a system. To achieve release privacy, privacy-preserving mechanisms such as data permutation, data perturbation or data anatomisation must be applied.

2.3 Privacy Notions, Models and Games and The Difficulty of Defining Privacy

By nature, privacy is multi-dimensional and multi-faceted, and can mean different things to different people. Therefore, 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. In our context of concern, 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 (*i.e.* 're-sanitisation' of a released (sanitised) version of a 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 that those algorithms became increasingly complex. Examples for non-interactive release privacy include [29,22,19].

To provide privacy in the interactive setting, the notion of *differential privacy* (DP) [10] emerged, and soon established itself as a *de facto* standard due to its mathematically rigorous definition and provable privacy guarantees. This powerful notion (or, more precisely, property of an algorithm) 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].

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. Examples include [2,5]. Such game-based definitions have been widely used in the cryptography community to provide simpler and more easily verifiable proofs. Similarly, via our games, we aim to clarify understanding of, and relationships between, different privacy notions.

3 A Distributed Privacy Model

Many privacy models (such as those of [29], [22], [19] and [10]) tend to assume that data to be shared originates from a centralised database. In this context, 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 — in which data remains split over several devices and is stored in databases *distributed* over a large network. Yet, constant improvements in technology allow increasingly more operations to be executed in distributed environments (which seek to preserve the privacy of those individuals involved in such processing). An example of such a system is a TRE [26], which operates in a distributed context and enables privacy-preserving

[2] Typical attacks targeting non-interactive mechanisms include linking attacks (for example, homogeneity attacks [22], skewness attacks [20] or similarity attacks [20]), composition attacks [15], minimality attacks [30] or the attack by deFinetti [16].

[3] Contributions that present criticisms and relaxations of the pure mode of DP are summarised in [9] and include (ϵ, δ)-DP [11], probabilistic (ϵ, δ)-DP [21], distributional privacy [6], computational DP [25], no-free lunch DP [17], personalised DP [13], and mutual-information DP [9].

release of data, while simultaneously ensuring verifiability (and therefore trustworthiness) of its algorithms. The following motivates the use of a distributed model, provides an overview of the privacy model of Bohli *et al.* [7] and elaborates on our model.

3.1 A Distributed Privacy Model

Systems that underlie a distributed model are assumed to store their data in many distributed databases. Yet, in our modelling, users gain access to the system via global `input` and `view` interfaces. Nevertheless, such a model captures use cases where data is distributed over several devices. Examples include deep learning [1], crowdsourcing statistics [14], federated learning[4] [24], or usage patterns collections (*i.e.* the approach by Apple[5] in iOS 10).

The benefits of such an approach are: (a) data remains at the data provider; (b) a reduced communication overhead; and (c) there is no single point of failure.

3.2 The Privacy Model of Bohli et al.

In their privacy model, Bohli *et al.* [7] do not explicitly state any assumptions pertaining to underlying (centralised or distributed) structure. 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, Bohli *et al.* give considerations to stateful, stateless and online systems.

For their system model, Bohli *et al.* 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. In particular, they define several notions for the properties of anonymity, unlinkability, pseudonymity and participation hiding, and show how these notions are related to each other.

3.3 Our Considerations: A Distributed Model

Cloud computing, the smart grid, collaborative learning (as well as others — see [4]) represent use cases that operate predominantly in distributed settings. These are all settings in which the deployment of privacy-preserving systems (such as a TRE) would tend to be beneficial. Consequently, to reason about privacy in such a setting, a privacy framework representing these needs is required. The extension of the model of [7], as presented in the following, was designed with a focus on the requirements of a distributed system. The model may be parameterised with any number of parties (*e.g.* $n = 1$ in case of centralised systems).

[4] https://research.googleblog.com/2017/04/federated-learning-collaborative.html
[5] https://wired.com/2016/06/apples-differential-privacy-collecting-data

We represent the notions of the model of [7] with privacy games. We extend the existing model by the notion of unobservability, which we, along with the other notions, relate to differential privacy. Furthermore, in our extension we give considerations to individual and group privacy. The notation in the sequel is consequently adapted from that of [7].

Informal Description of Privacy Notions The privacy notions considered in this paper are formally illustrated in Section 5, but we introduce them first in terms of natural language to aid comprehension.

Informally, we say that an adversary (simplified) is allowed to interact with a system in one of two ways:

(a) it inputs information, 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 of Bohli *et al.* [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).
- *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 to distinguish if a system invocation has taken place or not (by observing a real or 'generated' outcome element).

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*. To illustrate, consider the following examples: a breach of individuals' privacy would mean that an adversary is able, for any single element e_x a system releases, to infer its relation to the associated user u_x. Drawn from a genomics use case [4], e_x may represent a disease that user u_x has.

A breach of group privacy would mean that an adversary is able to relate two or more elements e_x, e_y, where e_x pertains to a user u_x, e_y pertains to a user u_y, and both users are member of a group $u_x, u_y \in \boldsymbol{u}_g$. In this case (assuming that e_x and e_y are types of diseases), the adversary may infer that e_x and e_y

pertains to a group of similar diseases. Another example would be: e_x is a specific disease, and if the adversary infers that $u_x, u_y \in \boldsymbol{u}_g$, then she may learn that u_y is somehow associated to e_x. More concretely, e_x is a specific inheritable disease and $\boldsymbol{u}_g$ is a family, with u_x and u_y members of that family — u_y, therefore, may also be susceptible to the disease. Hence, we model our notions and games such that they give consideration to both individual and group privacy.

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 a limited the buffer size of the system. 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.

4 System Model

We consider systems, denoted by ϕ^A, that are sequentially invoked a finite number of times, and require that they return values e_i and an aggregated value β_i computed over a finite number of inputs α_i. Further, we 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 $\boldsymbol{\alpha}_i$. The same applies for the output values e_i, with $\boldsymbol{e}_i$ 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^A are drawn from a parameter space $\alpha_i, e_i \in A$ that is specific to the system. For example, A may be the parameter space of $\mathbb{R}$. Similarly defined is the parameter space of the aggregated output $\beta_i \in A$. Each user identifier u_i and party identifier p_i is assumed to be unique, and drawn from an identifier space (for example, the parameter space $\mathbb{N}$ would be sufficient for these purposes).

Invocations of ϕ^A, in the form of a query q_i, produce output batches $(\boldsymbol{e}_i, \beta_i)$ of potentially varying size. Thus, a query q_i is essentially a mapping of input

batches to $(\boldsymbol{e}_i, \beta_i)$, denoted by π^{ϕ} for values $\boldsymbol{e}_i$ and π_{q_i} for the aggregated value β_i. In particular, ϕ^A accepts input batches of the form

$$(u_1, \alpha_1, p_1), (u_2, \alpha_2, p_2), \ldots, (u_c, \alpha_c, p_c)$$

for c invocations of the input oracle $\texttt{input}(\cdot, \cdot, \cdot)$. Furthermore, for each invocation of the output oracle, $\texttt{view}^{\pi_{\cdot}}(\cdot)$, ϕ^A applies a potentially secret permutation π^{ϕ} on the datasets held by each party p_i and outputs a sequence

$$(e_1, \ldots, e_c), \beta$$

where $\boldsymbol{e} \leftarrow \pi^{\phi}(\boldsymbol{\alpha})$ and $\beta \leftarrow \pi_{q_i}(\boldsymbol{\alpha})$.

In the notation 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 send her inputs. Our model is assumed to handle any data distribution along parties p_i.

Building on the privacy model of Bohli *et al.* [7], our model includes the possibility of a 'void' invocation of the system, *i.e.* an invocation of ϕ^A, where the outcome $(\boldsymbol{e}_i, \beta_i)$ 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^A. This property is essential in the context of system unobservability.

Similarly to Bohli *et al.*, we consider systems that are stateful, stateless or online. We take these properties into account when stating our privacy notions in Section 5 and modelling the privacy games in Section 6. Moreover, our stated notions and games must remain independent of the inputs and output of ϕ^A, and abstract to the mappings $\pi^{\phi} : \boldsymbol{\alpha}_i \rightarrow \boldsymbol{e}_i$ and $\pi_{q_i} : \boldsymbol{\alpha}_i \rightarrow \beta_i$.

Furthermore, our model allows single users or a collective of users to reason about their privacy assurances. Precisely, we do not limit a subject to contribute only a single element to ϕ^A, but allow for any finite number of entries. Thus, our notions and games deal with individual and group privacy. A notion or game can be applied to capture privacy properties about a mapping between:

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

where the last case denotes group privacy.

We denote the function f as the mapping between output values and user identifiers, and the function $f^{\star}$ as the mapping between dataset entries and user identifiers. More formally, $f : \boldsymbol{e}_i \rightarrow u_i$ takes as an input an element (or vector) e_i (or $\boldsymbol{e}_i$) of ϕ^A and maps it to the corresponding user u_i. Similarly, this holds for the internal processing of ϕ^A, $f^{\star} : \boldsymbol{\alpha}_i \rightarrow u_i$, with α_i (or $\boldsymbol{\alpha}_i$)

[6] Data submitted to the system may include (or, pertain to) several data subjects, that are independent of the actual data owner. In our modelling, so far, 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.

(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^A 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^A or the released outcome $(\boldsymbol{e}_i, \beta_i)$. Nevertheless, the adversary's goal remains to determine f, or any other interesting property about f. The privacy notions that we shall consider in Section 5 reveal, to varying degrees, information about f.

We denote $p_i \in P$, with P the set of all computational parties. Inputs of ϕ^A are stored in datasets $\alpha_i \rightarrow p_i.db$, given any possible distribution. Moreover, we say $\boldsymbol{\alpha} = \cup_{i=1}^{P} p_i.db$, that is, $\boldsymbol{\alpha}$ is the set of all inputs $\alpha_i \in \boldsymbol{\alpha}$ for all parties p_i. Similarly, it holds that $\boldsymbol{e} = \cup_{i=1}^{n} e_i$ represents the union of all outcome values. Following Bohli *et al.*, we consider the following properties of the mapping function f and ϕ^A. In Section 5, we will allow $\mathcal{A}$ to learn certain subsets of these properties, depending on the strength of the given notion.

Definition 1. *(Participant Set): $U_f = \{f(e_i) : e_i \in \boldsymbol{e}_u\}$ denotes the set of participants in ϕ^A. Each element $e_i \in \boldsymbol{e}$ 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 $\#u_i = |e_i \in \boldsymbol{e} : f(e_i) = u_i|$, denotes the relation of the number of elements $e_i \in \boldsymbol{e}$ 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^A.*

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

The right-hand side of Figure 2 illustrates the relationship of these sets.

5 Privacy Notions

We now state the notions for privacy that we consider in our privacy model. These definitions follow and extend the privacy notions of Bohli *et al.* [7]. In formulating said notions, we take into account that such a notion 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.* centralised or distributed), and must give consideration to individuals as well as groups.

Overall, the aim of $\mathcal{A}$ is to learn the mapping $f : \boldsymbol{e}_i \rightarrow u_i$ or any other interesting property about f. We formulate these 'properties' about f, from the viewpoint of the system ϕ^A, and give an informal description of the notion. Moreover, we state the conditions for each property, from the viewpoint of $\mathcal{A}$, illustrating the goal that $\mathcal{A}$ aims to learn in order to break said notion. Precisely, $\mathcal{A}$ is allowed to interact adaptively with the oracles of ϕ^A, and, at a given point in time, must commit an input value pair $(u_x, \boldsymbol{\alpha}_x)$ to which it receives the output

tuple $(\boldsymbol{e}_x, \beta_x)$ from ϕ^A, which we further denote as *general observation*. Given these parameters (and access to some system-specific oracles, as defined by the notions below) it will be evaluated if $\mathcal{A}$ is able to successfully break the given notion. More details of how $\mathcal{A}$ interacts with ϕ^A are given in Section 6.

Further, for individual privacy, we assume u_x to be fixed; for group privacy, we assume $u_x \in \boldsymbol{u}_g$, where $\boldsymbol{u}_g$ is a fixed group of user identifiers.

Definition 4. *(Strong Anonymity): A system is said to provide strong anonymity, denoted by* `SA`*, if it does not leak any information about the mapping function* f*.*

A system ϕ^A that satisfies the property of `SA` prevents $\mathcal{A}$ relating output elements to their associated user identifiers. More precisely, $\mathcal{A}$ must not be able, for any element $e_x \in \boldsymbol{e}_x$, to learn the associated identifier u_x, such that $f(e_x) = u_x$. $\mathcal{A}$ is not given anything beyond the general observation.

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

A system ϕ^A that satisfies the property of `PH` prevents $\mathcal{A}$ relating output elements to their associated user identifiers. More precisely, $\mathcal{A}$ must not be able, for any element $e_x \in \boldsymbol{e}_x$, to learn the associated identifier u_x such that $f(e_x) = u_x$. In addition to the general observation, we assume that $\mathcal{A}$ is able to learn $|U_f|$, which might or might not help to find said relation.

Definition 6. *(Strong Unlinkability): A system is said to provide strong unlinkability, denoted by* `SU`*, if it does not leak any information about* f *beyond the participant set* U_f*.*

A system ϕ^A that satisfies the property of `SU` prevents $\mathcal{A}$ relating output elements pertaining 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 $e_x, e_y \in \boldsymbol{e}_x$, such that $f(e_x) = f(e_y) = u_x$. In addition to the general observation, we assume that $\mathcal{A}$ is able to learn U_f, which might or might not help to find said relation.

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

A system ϕ^A that satisfies the property of `WU` prevents $\mathcal{A}$ relating output elements pertaining 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 $e_x, e_y \in \boldsymbol{e}_x$ such that $f(e_x) = f(e_y) = u_x$. 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, which might or might not help to find said relation.

[7] Note that knowledge of Q_f implicitly implies knowledge of U_f.

Definition 8. *(Pseudonymity): A system is said to provide pseudonymity, denoted by* `PS`*, if it does not leak any information about f beyond the linking relation P_f.*

A system ϕ^A that satisfies the property of `PS` prevents $\mathcal{A}$ relating a group of output elements (or an element within that group), pertaining 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 $\boldsymbol{e}_{u_x} \subseteq \boldsymbol{e}_x$ such that $\forall e_x \in \boldsymbol{e}_{u_x} | f(e_x) = u_x$. In addition to the general observation, we assume that $\mathcal{A}$ is able to learn P_f. This allows $\mathcal{A}$ to assign, for each partition of the overall outcome set $\boldsymbol{e}_x$, a pseudonym, which might or might not help to find said relation.

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

A system ϕ^A that satisfies the property of `AN` prevents $\mathcal{A}$ relating a group of output elements (or an element within that group), pertaining 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 $\boldsymbol{e}_{u_x} \subseteq \boldsymbol{e}_x$ such that $\forall e_x \in \boldsymbol{e}_{u_x} | f(e_x) = u_x$. 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 $\boldsymbol{e}_x$, however not the relationship between those sets, which might or might not help to find said relation.

Definition 10. *(Weak Anonymity): A system is said to provide weak anonymity, denoted by* `WA`*, if it does not leak any information about f beyond the participation set U_f, the usage frequency set Q_f and the linking relation P_f.*

A system ϕ^A that satisfies the property of `WA` prevents $\mathcal{A}$ relating a group of output elements (or an element within that group), pertaining 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 $\boldsymbol{e}_{u_x} \subseteq \boldsymbol{e}_x$ such that $\forall e_x \in \boldsymbol{e}_{u_x} | f(e_x) = u_x$. 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 $\boldsymbol{e}_x$ and information about on the number of elements each user has contributed in the outcome set, which might or might not help to find said relation.

Remark 1. In the case that each user contributes a unique amount of elements to ϕ^A, 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 previously mentioned sets.

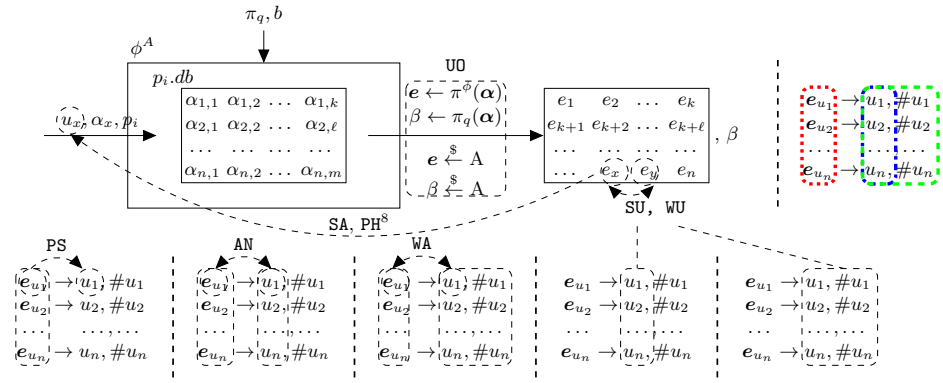

Fig. 2: Illustration of privacy notions, given access to the system ϕ^A, and oracle access to the sets $\mathbf{U_f}$ (- - -), $\mathbf{Q_f}$ (- - -) and $\mathbf{P_f}$ (- - -).

Definition 11. *(Unobservability): A system is said to be unobservable, denoted by* `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^A that satisfies the property of `UO` prevents $\mathcal{A}$ from distinguishing that, for a fixed tuple of user identifier and input value, a system invocation has taken place or not. More precisely, $\mathcal{A}$ must not be able, for a fixed input tuple (u_x, α_x), by observing the outcome $(\boldsymbol{e}_x, \beta_x)$, be able to distinguish if $\boldsymbol{e}_x \leftarrow \pi^{\phi}(\boldsymbol{\alpha}_x), \beta_x \leftarrow \pi_q(\boldsymbol{\alpha}_x)$ given $\alpha_x \in \boldsymbol{\alpha}_x, f^{\star}(\alpha_x) = u_x$, or, given $\alpha_x \notin \boldsymbol{\alpha}_x, f^{\star}(\alpha_x) \neq u_x$. Intuitively, it is easy to distinguish a system ϕ^A behaving in the above way, by simply determining $|\boldsymbol{e}_x|$. If $\alpha_x \notin \boldsymbol{\alpha}_x$, then $|\boldsymbol{e}_x| < |\boldsymbol{e}_y|$, where in the latter case $\alpha_x \in \boldsymbol{\alpha}_x$. Therefore, even for a *void* invocation of ϕ^A, there must hold $\exists e_x \in \boldsymbol{e}_x | e_x \not\leftarrow \pi^{\phi}(\alpha_x)$. e_x must be produced in a way that is indistinguishable of a system being invoked or not. 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 $\boldsymbol{e}_x$ and information about the number of elements each user has contributed in the outcome set, which might or might not help to find said relation.

In Figure 2 we illustrate previously defined privacy notions. As depicted, the system ϕ^A is queried with a triple (u_x, α_x, p_i) and releases a tuple $(\boldsymbol{e}, \beta)$ 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.

[8] For the notion of `PH`, $\mathcal{A}$ has additional access to an oracle giving the size of the participation set $|\mathbf{U_f}|$.

6 Adversarial Model and Privacy Games

Given the definition of the privacy notions for a distributed privacy model, we are now able to derive games expressing said notions. We begin by defining the environment and restrictions that apply when using the game-based technique. We then present diverse privacy games and, as a means of illustration, discuss the execution flow of the games.

6.1 Fundamentals

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 oracles, as defined by the privacy game. An oracle, denoted by $\mathcal{O}$, gives $\mathcal{A}$ access to certain knowledge about ϕ^A and relations between the outcome values $(\boldsymbol{e}, \beta)$. The interaction of $\mathcal{A}$ with an oracle $\mathcal{O}$ is further denoted by $\mathcal{A}^{\mathcal{O}}$.

Overall, we say that $\mathcal{A}$ is interested in learning the mapping function f, or any interesting properties about f, represented by the previously defined notions. 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 to `true` with a non-negligible probability. The advantage of $\mathcal{A}$ is denoted by

$$\mathbf{Adv}(\mathcal{A}) = Pr[G_{\star}^{\mathcal{A}} \rightarrow \texttt{true}]$$

where $\star \in \{\texttt{SA}, \texttt{PH}, \texttt{SU}, \texttt{WU}, \texttt{PS}, \texttt{AN}, \texttt{WA}, \texttt{UO}\}$ denotes any of the games.

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^A (*e.g.* it does submit a random value $e_x \notin \boldsymbol{e}$, where $\boldsymbol{e} \leftarrow \mathcal{A}^{\texttt{view}^{\pi_q}()}$), or any of the other oracles $\mathcal{O}_{U_f}, \mathcal{O}_{Q_f}, \mathcal{O}_{P_f}$ or $\mathcal{O}_{|U_f|}$, revealing specific information about certain subsets of f.

These restrictions allow us to abstract (and in some cases omit) some of ϕ^A'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 validation procedures) for any of the values submitted to the oracles. Moreover, we omit the definition of the oracles $\mathcal{O}_{U_f}, \mathcal{O}_{Q_f}, \mathcal{O}_{P_f}$ or $\mathcal{O}_{|U_f|}$, as we see such definitions as a technicality.

Any game $G_{\star}$ consists of a limited number of procedures. A procedure, denoted by **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 **proc** $\star$, where $\star \in \{\texttt{SA}, \texttt{PH}, \texttt{SU}, \texttt{WU}, \texttt{PS}, \texttt{AN}, \texttt{WA}, \texttt{UO}\}$.

In the following, 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,

proc initialise
$c \leftarrow 0$;
forall i in range$(0, n)$: $p_i.db \leftarrow \emptyset$

proc$_{\circlearrowleft}$ input(u_i, α_i, p_i)
$p_i.db \leftarrow p_i.db \cup (u_i, \alpha_i)$
$c \leftarrow c + 1$

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

proc$_{\circlearrowleft}$ corrupt(p_i)
return $p_i.db$

Fig. 3: Game G.

proc validate$_{SA}(e_x, u_x)$
if $f(e_x) = u_x$ then: return `true` else:
return `false`

proc SA
$\mathcal{A}^{\texttt{input}(u_x, \alpha_x, p_x)}$
$\boldsymbol{e}_x, \beta_x \leftarrow \mathcal{A}^{\texttt{view}^{\pi_q}()}$
return $\mathcal{A}^{\texttt{validate}_{\texttt{SA}}(e_x, u_x)}$

Fig. 4: Game $G_{\texttt{SA}}$ (Strong Anonymity).

call any of the procedures of G_y. In our case: $G_\star \vDash G$. In Figure 3, we present game G. $\mathcal{A}$ may play game G for an *arbitrary* number of times, before it plays a specific game $G_\star$ *once*, where she aims to break the expressed notion.
In general, $\mathcal{A}$ interacts with ϕ^A ('plays' the games) in the following way:

1. $\mathcal{A}$ inputs data (α_i) into the system ϕ^A.
2. $\mathcal{A}$ obtains[9] output $(\boldsymbol{e}, \beta)$ from the system ϕ^A.
3. $\mathcal{A}$ selects any value from the output, to which it believes it knows the relation (specified by the privacy notions) and submits that to the validation oracle (*e.g.* any $e_x \in \boldsymbol{e}_x$, where $\mathcal{A}$ believes to know u_x).
4. Game $G_\star$ returns `true` or `false` according to the output of the validation oracle.

As already mentioned, our privacy model operates in a distributed setting, but it also supports a centralised setting. For the games, we define an integer n, which represents the number of parties $p \in P$, with P the set of all parties. Each party p_i holds a database, denoted as $p_i.db$, containing the input values submitted to it. Then, for a centralised model, we say $n = 1$ (meaning we have exactly one party), and for a distributed model $n > 1$.

6.2 Privacy Games

In Figures 4-11, we illustrate the games $G_\star$, with $\star \in \{\texttt{SA}, \texttt{PH}, \texttt{SU}, \texttt{WU}, \texttt{PS}, \texttt{AN}, \texttt{WA}, \texttt{UO}\}$. Most of the games are quite similar, and differ only in the validation oracle and by the access to the oracles $\mathcal{O}_{U_f}, \mathcal{O}_{Q_f}, \mathcal{O}_{P_f}$ or $\mathcal{O}_{|U_f|}$. Therefore, we can group certain games and notions, as they aim to achieve similar objectives. In particular, we group the notions as follows: (`SA`, `PH`), (`SU`, `WU`), (`PS`, `AN`, `WA`) and (`UO`).

[9] $\mathcal{A}$ obtains $(\boldsymbol{e}, \beta)$ by querying the `view` oracle. For each party's database a permutation is applied. Moreover, $\boldsymbol{e}_x \circ \boldsymbol{e}_y$ denotes the combination of two sets $\boldsymbol{e}_x, \boldsymbol{e}_y$ under this permutation. This means that $\pi^{\phi}(\boldsymbol{e}_x) \circ \pi^{\phi}(\boldsymbol{e}_y)$ is equivalent to $\pi^{\phi}(\boldsymbol{e}_x \cup \boldsymbol{e}_y)$. Similarly, for β, $\circ$ denotes the π_q operation.

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

Fig. 5: Game $G_{\texttt{SU}}$ (Strong Unlinkability).

$\underline{\textbf{proc}_{\circlearrowleft}\ \mathcal{O}_{\mathbf{U_f}}}$
return U_f
$\underline{\textbf{proc}_{\circlearrowleft}\ \mathcal{O}_{\mathbf{P_f}}}$
return P_f
$\underline{\textbf{proc validate}_{PS}(\boldsymbol{e}_{u_x}, u_x)}$
if $f(\boldsymbol{e}_{u_x}) = u_x$ then: return `true` else:
return `false`
$\underline{\textbf{proc PS}}$
$\mathcal{A}^{\texttt{input}(u_x, \alpha_x, p_x)}$
$\boldsymbol{e}_x, \beta_x \leftarrow \mathcal{A}^{\texttt{view}^{\pi_q}()}$
return $\mathcal{A}^{\texttt{validate}_{\texttt{PS}}(\boldsymbol{e}_{u_x}, u_x)}$

Fig. 6: Game $G_{\texttt{PS}}$ (Pseudonymity).

$\underline{\textbf{proc}_{\circlearrowleft}\ \mathcal{O}_{\mathbf{U_f}}}$
return U_f
$\underline{\textbf{proc}_{\circlearrowleft}\ \mathcal{O}_{\mathbf{Q_f}}}$
return Q_f
$\underline{\textbf{proc}_{\circlearrowleft}\ \mathcal{O}_{\mathbf{P_f}}}$
return P_f
$\underline{\textbf{proc validate}_{UO}(g)}$
return $(g == b)$

$\underline{\textbf{proc}_{\circlearrowleft}\ \textbf{view}^{\pi_q}_{\textbf{refut.}}}$
$b \xleftarrow{\$} \{0, 1\}$
forall i in range$(0, n)$:
$\boldsymbol{e}_i \leftarrow \pi^{\phi}(p_i.db); \beta_i \leftarrow \pi_q(pi.db)$
if $b = 0$ then: $\boldsymbol{e}_{u_x} \xleftarrow{\$} A; \beta_{u_x} \xleftarrow{\$} A$
$\boldsymbol{e} \leftarrow \boldsymbol{e}_1 \circ \boldsymbol{e}_2 \circ \cdots \circ \boldsymbol{e}_n$
$\beta \leftarrow \beta_1 \circ \beta_2 \circ \cdots \circ \beta_n$
return $(\boldsymbol{e}, \beta)$

$\underline{\textbf{proc UO}}$
$\mathcal{A}^{\texttt{input}(u_x, \alpha_x, p_x)}$
$\boldsymbol{e}_x, \beta_x \leftarrow \mathcal{A}^{\texttt{view}^{\pi_q}_{\textbf{refut.}}()}$
return $\mathcal{A}^{\texttt{validate}_{\texttt{UO}}(g)}$

Fig. 7: Game $G_{\texttt{UO}}$ (Unobservability).

(a) **Games relating to single elements:** The objective of group (`SA`, `PH`) is, from the outcome $\boldsymbol{e}_x, \beta_x$, to find any value $e_x \in \boldsymbol{e}_x$ for which $\mathcal{A}$ is able to identify the user identifier u_x. The validation functions of this group return `true` if $\mathcal{A}$ is able to do so.

(b) **Games relating to multiple elements:** The objective of group (`SU`, `WU`) is, from the outcome $\boldsymbol{e}_x, \beta_x$, to find any two values $e_x, e_y \in \boldsymbol{e}_x$ for which $\mathcal{A}$ believes they are related to the user identifier u_x. The validation functions of this group return `true` if $\mathcal{A}$ is able to do so.

(c) **Games relating to element groups:** The objective of group (`PS`, `AN`, `WA`) is, from the outcome $\boldsymbol{e}_x, \beta_x$, given the ability to group certain outcome elements together, for this very group $\boldsymbol{e}_{u_x}$, to identify the user identifier u_x. The validation functions of this group return `true` if $\mathcal{A}$ is able to do so.

(d) **Games relating to system behaviour:** Finally, the objective group (`UO`) is, from the outcome $\boldsymbol{e}_x, \beta_x$, to distinguish if the values $\boldsymbol{e}_{u_x}, \beta_{u_x}$ were generated as a result of a system invocation, or generated purely at random. The validation functions of this group return `true`, if $\mathcal{A}$ is able to do so.

Remark 2. The previous illustration of the privacy games is for individual privacy, where we assume u_x to be fixed. For group privacy, we assume $u_x \in \boldsymbol{u}_g$, where $\boldsymbol{u}_g$ is a group of fixed user identifiers. Then, each objective may be extended to find elements $e_x \in \boldsymbol{e}_x$ that relate to any u_x pertaining to the group of $\boldsymbol{u}_g$. 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 $u_x, u_y \in \boldsymbol{u}_g$, then this would violate group privacy.

7 Conclusions

We set out to provide a clear and concise understanding of various notions for privacy-preserving data release. Precisely, we build upon the notions of other authors and formulate those notions into privacy games, following the game-playing technique of provable security. Furthermore, we extend the framework of Bohli *et al.* [7] by the notion of unobservability — a notion that captures the capability of an adversary to distinguish if a system invocation has taken place or not. 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). With the definition of the privacy notions and games, we take into consideration the different requirements of privacy protections for individuals and groups. Overall, many currently 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 of privacy aspects between the release of various systems), policies for the selection of privacy notions, exploration of primitives guaranteeing such notions and, most importantly, application to real-world case studies.

References

1. Abadi, M., Chu, A., Goodfellow, I., McMahan, H.B., Mironov, I., Talwar, K., Zhang, L.: Deep learning with differential privacy. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. pp. 308–318. CCS '16, ACM, New York, NY, USA (2016)
2. Achenbach, D., Huber, M., Müller-Quade, J., Rill, J.: Closing the Gap: A Universal Privacy Framework for Outsourced Data, pp. 134–151. Springer International Publishing, Cham (2016)
3. Anati, I., Gueron, S., Johnson, S., Scarlata, V.: Innovative Technology for CPU based Attestation and Sealing. In: Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy. HASP 2013, vol. 13. ACM (2013)
4. Ankele, R., Küçük, A., Martin, A., Simpson, A., Paverd, A.: Applying the trustworthy remote entity to privacy-preserving multiparty computation: Requirements and criteria for large-scale applications. In: 2016 Intl IEEE Conferences on Ubiquitous Intelligence Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld). pp. 414–422. IEEE (July 2016)
5. Bellare, M., Rogaway, P.: The game-playing technique. Cryptology ePrint Archive, Report 2004/331 (2004)
6. Blum, A., Ligett, K., Roth, A.: A learning theory approach to non-interactive database privacy. In: Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing. pp. 609–618. STOC '08, ACM, New York, NY, USA (2008)

7. Bohli, J.M., Pashalidis, A.: Relations among privacy notions. ACM Trans. Inf. Syst. Secur. 14(1), 4:1–4:24 (Jun 2011)
8. Corporation, I.: Intel Software Guard Extensions Programming Reference (2014), https://software.intel.com/sites/default/files/managed/48/88/329298-002.pdf
9. Cuff, P., Yu, L.: Differential privacy as a mutual information constraint. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. pp. 43–54. CCS '16, ACM, New York, NY, USA (2016)
10. Dwork, C.: Differential privacy. In: Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II. pp. 1–12 (2006)
11. Dwork, C., Kenthapadi, K., McSherry, F., Mironov, I., Naor, M.: Our data, ourselves: Privacy via distributed noise generation. In: Proceedings of the 24th Annual International Conference on The Theory and Applications of Cryptographic Techniques. pp. 486–503. EUROCRYPT'06, Springer-Verlag, Berlin, Heidelberg (2006)
12. Dwork, C., Roth, A.: The algorithmic foundations of differential privacy. Found. Trends Theor. Comput. Sci. 9(3), 211–407 (Aug 2014), http://dx.doi.org/10.1561/0400000042
13. Ebadi, H., Sands, D., Schneider, G.: Differential privacy: Now it's getting personal. In: Proceedings of the 42Nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. pp. 69–81. POPL '15, ACM, New York, NY, USA (2015)
14. Erlingsson, U., Pihur, V., Korolova, A.: Rappor: Randomized aggregatable privacy-preserving ordinal response. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. pp. 1054–1067. CCS '14, ACM, New York, NY, USA (2014)
15. Ganta, S.R., Kasiviswanathan, S.P., Smith, A.: Composition attacks and auxiliary information in data privacy. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 265–273. KDD '08, ACM, New York, NY, USA (2008)
16. Kifer, D.: Attacks on privacy and definetti's theorem. In: Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data. pp. 127–138. SIGMOD '09, ACM, New York, NY, USA (2009)
17. Kifer, D., Machanavajjhala, A.: No free lunch in data privacy. In: Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data. pp. 193–204. SIGMOD '11, ACM, New York, NY, USA (2011)
18. Küçük, A., Paverd, A., Martin, A., Asokan, N., Simpson, A., Ankele, R.: Exploring the Use of Intel SGX for Secure Many-Party Applications. pp. 5:1–5:6. SysTEX 2016, ACM (2016)
19. Li, N., Li, T., Venkatasubramanian, S.: t-Closeness: Privacy Beyond k-Anonymity and l-Diversity. In: Proceedings of the 23rd IEEE International Conference on Data Engineering (2007)
20. Li, N., Li, T., Venkatasubramanian, S.: t-closeness: Privacy beyond k-anonymity and l-diversity. In: 2007 IEEE 23rd International Conference on Data Engineering. pp. 106–115 (April 2007)
21. Machanavajjhala, A., Kifer, D., Abowd, J., Gehrke, J., Vilhuber, L.: Privacy: Theory meets practice on the map. In: Proceedings of the 2008 IEEE 24th International Conference on Data Engineering. pp. 277–286. ICDE '08, IEEE Computer Society, Washington, DC, USA (2008)
22. Machanavajjhala, A., Kifer, D., Gehrke, J., Venkitasubramaniam, M.: L-diversity: Privacy Beyond K-anonymity. ACM Trans. Knowl. Discov. Data (2007)

23. McKeen, F., Alexandrovich, I., Berenzon, A., Rozas, C.V., Shafi, H., Shanbhogue, V., Savagaonkar, U.R.: Innovative Instructions and Software Model for Isolated Execution. In: Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy (2013)
24. McMahan, H.B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS) (2016), http://arxiv.org/abs/1602.05629
25. Mironov, I., Pandey, O., Reingold, O., Vadhan, S.: Computational differential privacy. In: Proceedings of the 29th Annual International Cryptology Conference on Advances in Cryptology. pp. 126–142. CRYPTO '09, Springer-Verlag, Berlin, Heidelberg (2009)
26. Paverd, A.: Enhancing Communication Privacy Using Trustworthy Remote Entities. D.Phil thesis, University of Oxford (2016)
27. Pfitzmann, A., Hansen, M.: A terminology for talking about privacy by data minimization: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management (2009)
28. Pfitzmann, A., Köhntopp, M.: Anonymity, Unobservability, and Pseudonymity — A Proposal for Terminology, pp. 1–9. Springer Berlin Heidelberg, Berlin, Heidelberg (2001)
29. Sweeney, L.: K-anonymity: A Model for Protecting Privacy. Int. J. Uncertain. Fuzziness Knowl.-Based Syst. (2002)
30. Wong, R.C.W., Fu, A.W.C., Wang, K., Pei, J.: Minimality attack in privacy preserving data publishing. In: Proceedings of the 33rd International Conference on Very Large Data Bases. pp. 543–554. VLDB '07, VLDB Endowment (2007)

A Additional Privacy Games

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

Fig. 8: Game $G_{\mathtt{PH}}$ (Participation Hiding).

proc$_{\circlearrowleft}$ $\mathcal{O}_{\mathbf{U_f}}$
  return U_f
proc$_{\circlearrowleft}$ $\mathcal{O}_{\mathbf{Q_f}}$
  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
  $\mathcal{A}^{\mathtt{input}(u_x, \alpha_x, p_x)}$
  $\mathcal{A}^{\mathtt{input}(u_x, \alpha_y, p_x)}$
  $\boldsymbol{e}_x, \beta_x \leftarrow \mathcal{A}^{\mathtt{view}^{\pi_q}()}$
  return $\mathcal{A}^{\mathtt{validate}_{\mathtt{WU}}(e_x, e_y, u_x)}$

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

proc$_{\circlearrowleft}$ $\mathcal{O}_{\mathbf{U_f}}$
  return U_f
proc$_{\circlearrowleft}$ $\mathcal{O}_{\mathbf{P_f}}$
  return P_f
proc validate$_{AN}(\boldsymbol{e}_{u_x}, u_x)$
  if $f(\boldsymbol{e}_{u_x}) = u_x$ then: return `true` else:
    return `false`
proc AN
  $\mathcal{A}^{\mathtt{input}(u_x,\alpha_x,p_x)}$
  $\boldsymbol{e}_x, \beta_x \leftarrow \mathcal{A}^{\mathtt{view}^{\pi_q}()}$
  return $\mathcal{A}^{\mathtt{validate}_{\mathtt{AN}}(\boldsymbol{e}_{u_x},u_x)}$

Fig. 10: Game $G_{\mathtt{AN}}$ (Anonymity).

proc$_{\circlearrowleft}$ $\mathcal{O}_{\mathbf{U_f}}$
  return U_f
proc$_{\circlearrowleft}$ $\mathcal{O}_{\mathbf{Q_f}}$
  return Q_f
proc$_{\circlearrowleft}$ $\mathcal{O}_{\mathbf{P_f}}$
  return P_f
proc validate$_{WA}(\boldsymbol{e}_{u_x}, u_x)$
  if $f(\boldsymbol{e}_{u_x}) = u_x$ then: return `true` else:
    return `false`
proc WA
  $\mathcal{A}^{\mathtt{input}(u_x,\alpha_x,p_x)}$
  $\boldsymbol{e}_x, \beta_x \leftarrow \mathcal{A}^{\mathtt{view}^{\pi_q}()}$
  return $\mathcal{A}^{\mathtt{validate}_{\mathtt{WA}}(\boldsymbol{e}_{u_x},u_x)}$

Fig. 11: Game $G_{\mathtt{WA}}$ (Weak Anonymity).