Privacy in Revealed and Dependent Databases

Adam Bender

Master's Degree Scholarly Work

Abstract. This paper extends the recent work of Dwork, et.al., in several ways. We provide new definitions of varying levels of privacy, and prove their equivalence to a previous definition. We also prove two previous notions to be equivalent. We then initiate the first formal analysis of privacy in databases whose rows are dependent on each other.

1 Introduction

Privacy has long been an important topic in databases research, and has recently seen a resurgence in popularity among the computer science theory community. In this paper, we attempt to expand and generalize previous definitions of privacy, show their equivalence under certain assumptions, and begin the first undertaking of defining privacy for databases whose rows are not independent of each other.

Privacy in databases is a concern that is growing more important every day. As all sorts of records are increasingly being stored and managed digitally, their sheer volume mandates that they are stored in database systems. Along with the obvious advantages of easy search and retrieval, there are other benefits to this shift toward electronic record management. When medical records, credit card histories, and employment records are kept in databases, new methods of data mining [AS94] become possible, allowing researches and businesses alike to learn trends from the data. However, there has also arisen a competing concern. Often individuals' sensitive data is stored in these databases, and thus protecting the contents is a principal interest of these individuals. For instance, doctors and health professionals want to be able to query a database of AIDS patients so that trends that further research can be observed. On the other hand, patients do not want their personal information to be publicly available to any doctor in the world. The struggle between these two opposing positions is what motivates current research in database privacy - to find a method that both facilitates learning and protect privacy.

In order to study privacy, one must first define it. For a long time, the term "privacy" had evaded a universally-agreeable definition. Each subsequent paper redefined it in the terms that the authors saw most fit. It was hard to capture such an intrinsic notion in concrete terms; some definitions focused on prevention from being brought to the attention of others [CDM$^+$05], while others on preventing "too much" information from being learned about any individual record [DN03]. Privacy is also often confused with anonymity. The difference between the two notions is best highlighted by an example: let there be two people, Alice and Bob. Alice would like to send a letter to Bob. If Alice sends her letter without a return address and without signing it, then Alice is said to be anonymous in the sense that no one can trace the letter to her. The contents of the letter may become known, but the sender is not. However, if Alice were to lock her message in a box before sending it, so that only Bob can open it, the message is considered to be private. It can be known that Alice sent it, but the contents are known only to Alice and Bob. Anonymity is a term that refers to individuals, while privacy is a term that refers to information. Thus, research in this area has focused on privacy rather than anonymity.

Recently, Dinur, Nissim, Dwork, and others began one of the first formal studies of privacy, set in the framework of theoretical computer science. Their definitions focus on the notion of *confidence*, i.e., the certainty that someone interacting with a database has about the actual values of the records. This work follows in the same vein. We examine previous definitions of privacy in databases, and generalize them to show that many intermediate notions of privacy are possible. Then, we show equivalences between some of the previous definitions of privacy, and our new definitions. We then go on to give the first formal treatment of privacy in databases where the rows are dependently distributed, a topic that has been and continues to be difficult to work with in a formal setting.

1.1 Previous Work

Dinur and Nissim [DN03] initiated a recent surge in database privacy work, which includes [DN04, BDMN05, CDM$^+$05]. Their models provide a foundation on which many results have been built. We discuss them informally here.

Databases. Databases are, as their name suggests, repositories where data is stored, along with a method of performing computations on the data and reporting the results. These computations are commonly known as queries. A database can be thought of as a collection of n records, which are sometimes referred to as rows. Initial works focused on modeling rows as just a single bit and databases as n-bit vectors [DN03]. This definition was later extended to include rows composed of k bits, which gives the model of a database as a binary matrix [DN04]. Later, rows were again generalized to length-k vectors of real numbers, which can be treated as points in $\mathbb{R}^k$ (k-dimensional space) [CDM$^+$05]. Databases in this definition would then be $n \times k$ real matrices. Databases composed of arbitrary objects are considered in [BDMN05].

Queries. A database needs a way to retrieve the data that is stored inside of it. This is done through a query, which is a request for specific information that is stored in the database. Many previous works have only considered one type of query, which we also use in this paper. The type of query we are concerned with is a *noisy sum* query, as defined in [DN03, DN04, BDMN05]. This type of query to the database has two parameters: Q, a subset of the rows r of the database over which the query is performed, and a function $f : r \to \{0, 1\}$ mapping rows to 0 or 1, which is to be applied to the individual rows themselves. The result of the query is $\sum\limits_{r \in Q} f(r)$, or the number of rows for which the function f returns 1.

Adversaries. An adversary is a common notion in cryptography and security literature. It was recently extended to privacy literature. In adversary-based definitions of security, or in this case privacy, an *adversary* is an arbitrary human (or Turing machine) that attempts to defeat the integrity of a system by whatever means it has at its disposal. The abstract notion of security is given in terms of this adversary - results are of the form "an adversary that is limited by condition X cannot defeat this system." Analyzing the security of a real database would be a near-impossible task. Instead, theoretical models of databases (such as the one above) and related components are created, and the interaction between an attacker and a real database is modeled as a *game* between an adversary $\mathcal{A}$ and a database $\mathcal{D}$. This allows an analysis of privacy in much more general terms, as opposed to analyzing every single different query an adversary could make to a real database. $\mathcal{A}$'s goal in the game is to breach the privacy of $\mathcal{D}$. The game is designed so that if $\mathcal{A}$ cannot win the game, then he has no hope of breaching the privacy of a real database. In the game, $\mathcal{A}$ will issue queries to $\mathcal{D}$, then use the responses it receives, along with data it might have learned from either external sources or from the database itself, to breach the privacy of the database. For a game with an adversary, the adversary's specific goal conditions (i.e., what constitutes a loss of privacy) must be defined. In general, the adversary wins if it gains some restricted information from the database. That is, it wins if there is some hypothesis that *a posteriori* interacting with the database it does not know whether it is true or not, but *a priori* interacting it does. Obviously, if the adversary can learn the value of a record, it can verify many such hypotheses.

Without any sort of privacy protection, it is clear that $\mathcal{A}$ can easily win any such game: $\mathcal{A}$ picks some record in the database that it is not sure of, say a record r. Thus, *a priori* it has no information about r. It then makes a query to retrieve the contents of r (if there are restrictions on the type of query allowed, such as those defined above, then multiple queries might be necessary; but in fact even if the adversary learns *one bit* if new information about r, then it is considered to have won, so one query suffices). Now that it knows the contents of the row, it has *a posteriori* learned information from the database, and has won the game.

In order to prevent an adversary from winning a game so easily, many schemes used to protect privacy have been developed and studied in the past. In this paper, we focus on a single method called *output perturbation*. For a comprehensive study of other privacy protection methods, see [AW89, CDM$^+$05].

Output Perturbation. Output perturbation is the process of adding random noise to the response of a query. This is done to mask the true response, which as shown above would give (too much) information to the adversary. A perfectly privacy-preserving method would be to answer every query with random noise, but this clearly does not allow any sort of data mining. Instead, the true response and the random noise are added together, with the goal that, over many queries, some learning of general trends is possible, but it is not possible to learn enough about any single record.

One other common method of privacy preservation is to add random noise before publishing a database, and not alter the query responses in any way. This is called *database sanitization*. There are some advantages to output perturbation over database sanitization, which is why it is the method considered in this paper. For one, the original data is not altered. This means that if perfect learning were possible, using output perturbation would allow the real database to be learned, while using sanitization would allow only a modified database to be learned. This, more accurate learning can be done with output perturbation. Also, the amount of perturbation is variable. Certain "privilege levels" can be assigned to queries, so that a higher-privileged query undergoes less perturbation than a low-privilege query. In a sanitized database, the amount of perturbation is fixed.

2 Preliminaries

We adopt many of our basic definitions and constructions from the previously cited works, and present them here.

Database. We define a database to be an $n \times k$ matrix with entries in $\mathbb{R}$. Each of the n rows are assumed to correspond to a different record; each record has k real-valued attributes. Each row can be treated as a point in $\mathbb{R}^k$-space, by interpreting each of the k values as a distance along a certain dimension. The i'th row of a database $\mathcal{D}$ is referred to as $\mathcal{D}_i$.

Queries. A query is a question that is asked to the database, which then answers by examining its records. This work focuses on a specific type of query called a noisy sum, or statistical, query.

Definition 1 (Statistical query). *A* s*tatistical query is a query to a database that takes two user-supplied parameters: a set $\mathcal{Q}$ of rows, and a function $f : \mathbb{R}^k \to \{0, 1\}$. The result is $\sum_{r \in \mathcal{Q}} f(r) = |\{r \in \mathcal{Q} \mid f(r) = 1\}|$. That is, the function maps each row to either 0 or 1, and the result is the number of rows that map to 1.*

At first glance, this does not resemble the common notion of a database query, which when written in a language such as SQL would look something like this:

```
SELECT count(*)
```

```
FROM table
WHERE age >= 20 AND age <= 35
```

A seemingly more suitable definition of a query would be to replace the set Q with a k-hypercube. Then, interpret each row of the database as a point in $\mathbb{R}^k$. If a point is inside the hypercube, it is passed to the function f and potentially counted in the sum, otherwise it is not. Thus, instead of saying "Perform a query on rows 1-37", a user can now say "Perform a query on all persons between 20 and 35 years old." However, this type of query can be emulated by a noisy-sum query. For instance, if a query should be restricted to to records of people between 20 and 35 only, the query that is used should be $(\mathcal{Q}, f')$ where $\mathcal{Q}$ is the entire database $(\mathcal{D})$ and $f' = (age >= 20) \wedge (age <= 35) \wedge f$ where age is interpreted as a column of $\mathcal{D}$ that represents a record's age. Then, f' will only be true for records that have an age value between 20 and 35 and for which f is true, which will be the same records counted when restricting the function f to only rows with age values between 20 and 35. Thus, noisy-sum queries are a super-set of hypercube-based queries.

Revealing. One aspect of a real database that could potentially be hard to model in a formal definition of privacy is that copies of information in the database can be obtained from some other source. Consider the following scenario: Alice's personal information is entered into a medical database. Alice then publishes this same information on her website. An adversary, Bob, then tries to breach the privacy of the medical database. He does not interact with the database, but demonstrates that he has learned information by giving the data he found on Alice's website. Clearly, this is not a breach of privacy, even though Bob knows some information about Alice. Yet, if Bob were playing a privacy game against the database, he would have won. Ideally, information gained from outside the interaction with the database should not count as a breach of database privacy. In order to model the possibility of observing the sensitive from data outside of the database, previous works have allowed another form of database interaction: revealing rows. An adversary can obtain the contents of a row, un-modified, by calling the function $reveal(i)$ on database $\mathcal{D}$, which returns $\mathcal{D}_i$. If a real-life adversary learns about a row i from an outside source (something other than a query), that is modeling in a game by having the adversary call $reveal(i)$. In our example, when Bob looked at Alice's website, he would be considered to have called $reveal$(Alice's row), since he learned that information from some source besides a query. $reveal()$ is used as a generic, theoretical function that encompasses the many real-world ways in which an attacker can learn about a database from outside sources.

Previous works have allowed only a different type of reveal function: $reveal(\neg i)$, which reveals all records in $\mathcal{D}$ *except* i [DN03, DN04, BDMN05]. This is done at a specific step in an adversarial game, described in Section 2.1. We use a different definition of $reveal()$, because being able to reveal row individually allows more fine-grained control over what the adversary is allowed to do. This leads to our new classifications of privacy schemes. There is no added time complexity in revealing all but one of the database rows one at a time versus all at once A call to $reveal(\neg i)$ returns a table of $n-1$ records, which would take $n-1$ steps to read. Each call to $reveal(i)$ takes unit time, so revealing $n-1$ rows this way takes the same amount of time.

Output perturbation. As mentioned before, output perturbation is used to preserve the privacy of a database. Here, we give a technical discussion. After the result of a query to a database is determined, if that database is using output perturbation then it adds random noise to the result before giving the response to the user that issued the query. Recall that without output perturbation, the response to a query $(\mathcal{Q}, f)$ is $\sum_{r \in \mathcal{Q}} f(r)$. With output perturbation, the response to $(\mathcal{Q}, f)$ is now $\sum_{r \in \mathcal{Q}} f(r) + N(0, v)$, where $N(m, v)$ is random noise with mean m and variance v [BDMN05]. The authors of [DN03] showed that $v = O(\sqrt{n})$ was necessary for adversaries that work in time $O(n)$. Otherwise, the adversary could query a single row $O(n)$ times, and the noise would cancel out enough to give the adversary a good estimate of that row's contents. For an adversary restricted to work in sub-linear time, [BDMN05] shows that less noise is necessary to guarantee privacy. Our results in this work are independent of the running time of the adversary. In regards to what types of learning are possible with a sub-linear number of noisy-sum queries to a database using output perturbation, [BDMN05] also shows that principal component analysis, k means clustering, the Perceptron Algorithm, and the ID3 Algorithm, are still possible.

2.1 Privacy

We now explicitly define our notion of privacy, in terms of a game between an adversary and a database. It resembles the one found in [DN04]. We assume that there is some predicate $g : \mathbb{R}^k \to \{0, 1\}$ that the adversary $\mathcal{A}$ wishes to learn about a row r in the database $\mathcal{D}$. If $\mathcal{A}$ is able to learn the value of $g(r)$, it is said to have won the game – that is, it has to learn only a single bit to breach the privacy of $\mathcal{D}$. If the adversary is able to win the game, then there is a chance that it will be able to breach the privacy of a real-world database, but if it is unable to, there is no realistic way for it to be able to learn private information from such a database - as long as the real database sufficiently resembles the theoretical model. The general structure of a game consists of several stages, as follows:

1. $\mathcal{A}$ issues queries to $\mathcal{D}$, and reveals rows of $\mathcal{D}$ if it has the capability
2. $\mathcal{A}$ selects the row i that it will attempt to learn about, and a predicate g
3. $\mathcal{A}$ issues queries to $\mathcal{D}$, and reveals rows of $\mathcal{D}$ if it has the capability
4. $\mathcal{A}$ indicates it is finished, and it guesses $g(\mathcal{D}_i)$. If it guess correctly, and it never called *reveal*(i), it wins, otherwise it loses.

Let $p_{q,r}^{i,g}$ be the probability that $\mathcal{A}$ correctly guesses the value of $g(\mathcal{D}_i)$ after making q queries to the database and revealing r rows. Then $p_{0,0}^{i,g}$ is the *a priori* probability that $\mathcal{A}$ can guess $g(\mathcal{D}_i)$ before interacting with $\mathcal{D}$ at all, based solely on knowledge of the distribution from which $\mathcal{D}_i$ was drawn. Previous works have given definitions of privacy in terms of *confidence*, which is a function of these probabilities, and the change in confidence between before interacting with the database and afterward. For the sake of simplicity do not use the same definition; however, ours is functionally equivalent.

Definition 2 ((δ, q, r)-privacy). *A database response mechanism is* (δ, q, r)-private *if for every database* $\mathcal{D}$ *of* n *rows, every row index* i*, every function* $g : \mathbb{R}^k \to \{0, 1\}$*, and every adversary* $\mathcal{A}$ *making at most* q *queries and revealing* r *rows,* $\Pr\left[\left|p_{q,r}^{i,g} - p_{0,0}^{i,g}\right| > \delta\right] < \frac{1}{n^c}$ *for any constant* c*.*

Informally, if the probability that an adversary guesses $g(\mathcal{D}_i)$ is the same after querying the database and revealing its rows as it was before, then the database is said to be private. If the adversary does not have any prior knowledge about the value of $g(\mathcal{D}_i)$, and thus the probability that it is able to guess the value correctly is $\frac{1}{2}$, then the above equation can be restated as: $\Pr\left[p_{q,r}^{i,g} > \frac{1}{2} + \delta\right] < \frac{1}{n^c}$.

2.2 CCA-2 Security

When reading [DN03] we were struck by how similar their definition of privacy was to the definition of *adaptive chosen-ciphertext attack* in the literature of cryptography. Upon further inspection, a correlation exists between how an adversary attempting to defeat the security of a cryptosystem interacts with its oracles, and how an adversary attempting to breach the privacy of a database interacts with the database.

An understanding of adaptive chosen-ciphertext attack, or CCA-2, is not necessary for the understanding of this paper, so we will give only a brief overview. In order to prevent a message, or *plaintext* from being read by adversaries, it can be encrypted with a key into a *ciphertext* before being transmitted. If the recipient knows how to undo the encryption (decrypt), he can read the original message. Decryption requires knowledge of the encryption key (in some cases, more is required). In a CCA-2 game, the adversary attempting to break the cryptosystem interacts with an *oracle*, which can be thought of as the administrator of the game. For each game, there is a fixed encryption key that the adversary is unaware of. The adversary is allowed two operations: $encrypt(m)$, which returns the ciphertext created by encrypting m with the key, and $decrypt(c)$, which returns the plaintext that was encrypted with the key to produce c. Thus, $decrypt(encrypt(m)) = m$. The oracle is what responds to each of these operations. The steps of a CCA-2 game are as follows:

1. The oracle picks a key
2. The adversary is allowed to call $encrypt()$ and $decrypt()$ arbitrarily
3. The adversary sends two messages, m_1 and m_2, to the oracle. The oracle responds with c, which is an encryption of one of the messages
4. The adversary is allowed to call $encrypt()$ and $decrypt()$ arbitrarily
5. The adversary says whether it was m_1 or m_2 that was encrypted in step 3

The adversary wins if it correctly guesses which message was encrypted and it never called $decrypt(c)$ [GB].

Non-adaptive chosen-ciphertext attack (CCA-1) is similar, except that the adversary cannot call $decrypt()$ in step 4. *Chosen-plaintext attack* (CPA) is similar, except that the adversary cannot call $decrypt()$ at all. A cryptosystem is called CCA-2 secure if it can withstand a CCA-2 attack; similarly for CCA-1 and CPA.

2.3 Adversaries

We draw the following parallels between the definitions of privacy with output perturbation and the definitions of cryptographic security given above:

- Encryption is analogous to querying. In both cases, the adversary requests the result of data that is passed to a function that is intended to obscure the data's true contents. In the case of output perturbation, it is obscured by adding random noise; in the case of encryption, it is encrypted with a secret key.
- Decryption is akin to revealing a row of the database. Both allow an adversary to view data that would otherwise be obscured (either by output perturbation or encryption). However, both operations disqualify the adversary from using their output as an example of something it has learned (i.e., the privacy adversary only wins if it does not call $reveal(i)$ and the cryptographic adversary only wins if it does not call $decrypt(c)$).
- Presenting two plaintexts to the encryption oracle corresponds to choosing which row of the database that the adversary will attempt to breach. This is how the adversary indicates its goal, and prevents the adversary from using decrypted or revealed information as something that will win the game.

Thus, we can formulate three definitions of privacy, that correspond to each of CPA, CCA-1, and CCA-2 security, and compare them to the most common previous definition of privacy. This section refers to the game presented in Section 2.1.

Definition 3 (Privacy without revealing). *A database has* privacy without revealing *if it is resilient to an an adversary that is allowed to only query in steps 1 and 3 of the game.*

Thus, the adversary does not have the ability to call *reveal*(). This corresponds to a chosen-plaintext attack, where the adversary does not have the ability to call *decrypt*().

Definition 4 (Privacy with prior revealing). *A database has* privacy with prior revealing *if it is resilient to an adversary that can call reveal() in step 1, but not in step 3.*

This corresponds to a non-adaptive chosen-ciphertext attack. Here, the adversary can potentially learn which row would be easier to compromise by learning the contents of other rows.

Definition 5 (Privacy with post revealing). *A database has* privacy with post revealing *if it is resilient to an adversary that can call reveal() in step 3, but not in step 1.*

This is the standard definition, as presented in [DN03, DN04]. It does not correspond to any previous definition of a cryptographic adversary. See Appendix A for a definition of cryptographic security that corresponds to such a privacy adversary.

Definition 6 (Privacy with prior and post revealing). *A database has* privacy with prior and post revealing *if it is resilient to an adversary that can call reveal() in steps 1 and 3.*

This corresponds to an adaptive chosen-ciphertext attack, where the adversary has access to the *decrypt*() function at all times. Note that since the adversary learns nothing by selecting a row i in step 2 (unlike in the definition of CCA-2, where it learns the ciphertext c), all queries made in step 3 can be made in step 1 (similarly for any ability to call *reveal*). Therefore, querying in step 3 is useless, and revealing in that step is also useless if the adversary is able to reveal in step 1. Thus privacy with prior and post revealing is equivalent to privacy with prior revealing.

Isolator. The above game implies a realistic, but perhaps unnecessary, restriction, namely that the adversary must know the index i of the row of the record whose privacy it is trying to breach (since it has to output an i in step 2). This restriction can be relaxed by considering a different type of adversary, known as an *isolator*. This adversary succeeds if at any time it outputs a point p in $\mathbb{R}^k$ where p is "near" one point in the database, but "far" from others.

Unfortunately, the only formal definition of an isolator is for a different method of privacy preservation [CDM+05]. It is given in the context of a sanitized database, where the data is altered before being published. The advantage of the isolator $\mathcal{A}$ is given in terms of the sanitized database (SDB), any auxiliary information z, any point i in the real, un-sanitized database, and a Turing machine $\mathcal{A}'$ running in the same time as $\mathcal{A}$. A SDB is δ-resilient to an isolator if $|\Pr[\mathcal{A}(SDB, z) \text{ isolates } i] - \Pr[\mathcal{A}'(z) \text{ isolates } i]| < \delta$. A point p is isolated if the adversary outputs a hypercube H such that p lies within H, but fewer than c other points do, for some system parameter c.

To deal with this discrepancy, we use a generalization of this definition. Specifically, define for each row i in $\mathcal{D}$ a probability $P^i_{q,r}$, which is the probability that the adversary can isolate the point i, after making q queries and revealing r rows. The adversary wins if $\exists i : Pr\left[\left|P^i_{q,r} - P^i_{0,0}\right|\right] < \delta$. Note that the adversary does not need to know the index i in order to win this game. This is a generalization of (c, t)-isolation as presented in [CDM+05] (ignoring the fact that the definition in [CDM+05] is in terms of a sphere, not a cube), in that it is no longer specific to a sanitized database.

3 Results

In this section we make the conventional assumption that all rows are independently distributed. All previous work has also made this assumption, which only serves to highlight the contributions in Section 4. Using this assumption, we prove equivalences between all of the adversaries defined previously. This seems counterintuitive, in that an adversary who can only query a database is just as powerful as an adversary that can query a database where it additionally knows the contents of all but one row. Thus, the lack of dependencies between rows become a significant distinction.

Theorem 1. *Privacy with post revealing is equivalent to privacy with no revealing.*

Proof sketch. The only distinction between privacy with post revealing and privacy with no revealing is in step 3. Let $p_r = p_{q,r}^{g,i}$ be the probability that an adversary $\mathcal{A}$ can guess $g(\mathcal{D}_i)$ after revealing r rows. Assume $\mathcal{A}$ is in step 3 of a game. Consider two scenarios: $\mathcal{A}$ makes a query on row a and row b, then reveals row b, versus $\mathcal{A}$ makes a query solely on row a. In the first scenario, if $f(b) = 1$ then the response is $1 + f(a) + N(0, v)$. If $f(b) = 0$ then the response is $f(a) + N(0, v)$. The adversary can then call $\mathit{reveal}(b)$ to determine $f(a) + N(0, v)$. In the second scenario, the response is $f(a) + N(0, v)$. Therefore, the adversary gains the same information about row a by issuing a single query and *not* revealing the row. If $\mathcal{A}$ queried more than two rows, it would have to reveal even more rows to gain the same information. Thus, revealing rows in step 3 does not affect $p_{q,r}^{g,i}$; namely, $p_{q+1,r}^{g,i} = p_{q+1,r+1}^{g,i}$. Since revealing in step 3 does not help the adversary, it is unnecessary, so privacy with post revealing and privacy with no revealing are equivalent. □

Likewise, the intuition that allowing revealing before selecting a row can somehow allow an adversary to select a row that is easier to breach is false.

Theorem 2. *Privacy with prior revealing is the same as privacy with post revealing.*

Proof sketch. The same argument above applies, as there are no proof steps that depend on the adversary being in step 3. Therefore, it can also be applied to step 1. This gives the desired result. □

Thus, by transitivity, all four definitions are equivalent. A further, somewhat surprising, result is that these adversaries are also equivalent to an isolator, which perhaps could serve to enhance the results of [CDM+05]. We now turn our attention to isolator adversaries. We claim that if a database is has privacy with prior revealing, than it is also resilient to an isolator (this is one direction of showing equality, if we could show that a database resilient to an isolator also has privacy with prior revealing, then they would be equal, however there is no evidence that this is the case).

Theorem 3. *Privacy with prior revealing implies resistance to an isolator.*

Proof sketch. Let $\mathcal{D}$ be a database able to provide privacy with prior revealing. Let $\mathcal{A}$ be an isolator attempting to breach the privacy of $\mathcal{D}$ by isolating a point. As shown before, step 3 is unnecessary for $\mathcal{A}$, and can be skipped. Thus steps 2 and 4 can be combined, and instead of naming a row in step 2, $\mathcal{A}$ simply produces a hypercube H. For each point i in H, if $P_{q,r}^{i} > \delta$, and $\mathcal{A}$ did not call $\mathit{reveal}(i)$, then $\mathcal{A}$ wins. Let $g(j)$ be a predicate such that $g(j) = 1$ if $\mathcal{A}$ is able to isolate point j and 0 otherwise. Note that if $P_{q,r}^{i} > \delta$, then $\Pr[g(i) = 1] > \delta$, whereas before interacting with the database, $\Pr[g(i) = 1] < \delta$ (by definition of an isolator). Thus, $\mathcal{A}$ has learned some predicate about an entry in a database that has privacy with prior revealing, which is a contradiction. Therefore $\mathcal{A}$ cannot isolate any points i.

□

We now focus on databases where the rows can have dependent relationships on one another.

4 Dependent Rows

Up until now, there has been no work that addresses databases whose records are not drawn independently. This is likely because of the extreme difficulty in categorizing the many types of possible relations the rows can have. Consider, for example, a database in which every record shares the same value for an important attribute. A real life example would be a database of AIDS patients. In this case, every record should be kept private. Even if names and social security numbers are stripped from the database, identification of records is still possible. For instance, Sweeney claims that 87% of the U.S. population is uniquely identifiable from only birthdate, zip code, and gender [Swe97]. Clearly, these would be useful attributes for data mining, so would reasonable be left intact in a medical database. In this case an adversary who calls *reveal* once on this database may not be able to win any previously defined privacy game, yet the privacy of the individuals in database is clearly comprised. These are the type of complicated situations that arise when dealing with databases with dependent rows.

Consider another example[1] of an employee database, where it is known that employee X makes the same as employee Y. We can determine X's salary if we know what Y's salary is. The purpose of using the *reveal* function is to formally manage exactly which information can be learned from sources other than queries. Here, unlike as with independently drawn records, learning about one row can give information about another row. Since such out-of-band information becomes more of a liability in this case, we must manage it more closely than in the case with independently drawn records.

From these examples, it becomes clear that the equivalences shown in the previous sections do not hold for dependent databases. Indeed, the revealing of a single row could give an adversary information about all of the rows, which is impossible in a database with independently drawn rows. In order to aid our analysis, we take into account rows that, when revealed, give information about the contents of other rows. We define such a *master row* as one that, when changed, has the possibility of changing other rows (in order to keep the database consistent – in the above example, if X's salary changes, Y's must as well). That is, any row that that has another row dependent upon it is a master row. A row can have more than one master row that it depends upon.

4.1 Sensitivity

The analysis of Dwork, McSherry, Nissim and Smith in [DMNS06] defines the *sensitivity* of a function f, $S(f)$ as how much the output of f can change when the input changes very little.

As an example, consider the case of noisy-sum query (before output perturbation). If the input (the database) changes very little (one record is altered), the output can change by at most 1. Here, f would be $\sum_{r\in\mathcal{D}} g(r)$, $g : \mathbb{R}^k \to \{0, 1\}$, as defined previously. Since only one record changes, at most one value $g(r)$ changes, and the sensitivity of f is 1. Therefore the sensitivity of a noisy-sum query is 1.

We extend this definition to apply to the *sensitivity of databases* as a measure of the dependencies between records. Informally, this is the greatest number of records that can change when a single master record is changed. In a database with independently drawn rows, the sensitivity of that database is 1: when a single row changes, there are no others that depend on it that would have to change. In the case of the database salaries where two are forced to be the same, the sensitivity of that database, $S(\mathcal{D})$, would be 2: if X's salary were changed, Y's would have to be as well. Thus, if it were possible to determine the sensitivity of a database beforehand, one would be able to calibrate the amount of noise that would be needed to preserve privacy, as given in [DMNS06]. In that work, the Laplace(λ) distribution is used to provide the random noise. It is distributed according to $\Pr(x) \propto e^{-|x|/\lambda}$. Their result can be used to show that the amount of noise necessary for protecting noisy sum queries on a

[1]Due to Nick Frangiadakis

database with sensitivity S is Laplace(S/δ). Now that it is known how much noise is necessary to use to perturb query responses, we must now determine how many rows of a database can be revealed, while still protecting privacy.

4.2 Revealing

Many complications arise when considering revealing rows of dependent databases. One important question is whether or not the adversary knows the relations between the rows before interacting with the database. It seems more prudent to assume that he does. A more nuanced question is whether or not the adversary knows which rows of a database would be the master rows. Here it also seems prudent to assume that this is public knowledge, but here it seems to depend more on the individual databases themselves. A pessimistic approach is often the wisest, yet those assumptions leave a bleak outlook for future research in this direction.

After analyzing the vastly differing amounts of revealing that is possible in the above two examples, it is clear that one cannot easily quantify how many rows of a database can be revealed before privacy is breached. It is our view that any definition of privacy in dependent databases must be orthogonal to revealing the contents of rows, and should focus on the amount of noise to add to queries instead. It is a saddening fact that when applied to actual databases with rows that are dependent upon one another, any definition of privacy cannot permit or prove privacy against any reveal operation. To see this, just consider the two above examples. Revealing a single row compromises the privacy of someone in the AIDS database, whereas unless the adversary knows which row corresponds to X and Y, $n-1$ rows are able to be revealed without breaching privacy. No general statements can be made about revealing in dependent databases, as there are situations where all four definitions in Section 2.3 are equivalent, and others where they are not. This conjecture also extends to the relationships that each row has with another. An adversary examining the salary database in the above example might not know that the salary of X must equal the salary of Y, but if this information were somehow revealed, it would give the adversary new insight on the structure of the database.

In spite of this negative result, we offer one additional result: If the adversary does not know which rows are master rows, then the probability that they are all revealed, after r reveals, is

$$\begin{aligned}
&\frac{\binom{n-m}{r-m}}{\binom{n}{r}} \\
=&\frac{(n-m)!r!(n-r)!}{(r-m)!(n-r)!n!} \\
=&\frac{r^{\underline{m}}}{n^{\underline{m}}} \\
<&\frac{r^{m}}{n^{\underline{m}}}
\end{aligned}$$

where $n^{\underline{m}} = n(n-1)(n-2)\cdots(n-m+1)$.

This can be used to set a limit on the number of rows that can be revealed before it is expected that the adversary can learn anything significant about the dependencies in the rows of the database.

5 Conclusion

In this paper, we build upon previous works in database privacy. We then go on to derive a precise taxonomy of privacy in databases, using definitions of adversaries from cryptography, which we extend to apply to privacy. In the case of databases with independently drawn rows, we show that each of these notions is equivalent to each other. In addition, the previous notion of an isolator adversary is also equivalent to these definitions, which might be helpful in bringing together the analysis of output perturbation (the setting in which our adversaries are defined) and database sanitization (the setting in which the isolator was originally defined). We have also initiated a study of privacy in databases where the rows can depend on one another. The results in this section are mixed: we show that if it is possible to determine the sensitivity of a database, the amount of noise that needed to mask a query can be determined. However, we give evidence of our belief that revealing rows in dependently-drawn databases cannot be generically quantified.

6 Acknowledgments

We would like to thank Nick Frangiadakis for asking pertinent questions and Jonathan Katz for guidance.

References

[AS94] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. In *VLDB*, 1994.

[AW89] Nabil R. Adam and John C. Worthmann. Security-control methods for statistical databases: a comparative study. *ACM Comput. Surv.*, 21(4):515–556, 1989.

[BDMN05] Avrim Blum, Cynthia Dwork, Frank McSherry, and Kobbi Nissim. Practical privacy: The sulq framework. In *PODS*, ACM, pages 128–138, 2005.

[CDM$^+$05] Shuchi Chawla, Cynthia Dwork, Frank McSherry, Adam Smith, and Hoeteck Wee. Toward privacy in public databases. In *2nd Theory of Cryptography Conference (TCC)*, pages 363–385, 2005.

[CS98] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *CRYPTO*, pages 13–25, London, UK, 1998. Springer-Verlag.

[DMNS06] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *3rd Theory of Cryptograph Conference (TCC)*, volume 3876 of *LNCS*, pages 265–284, 2006.

[DN03] Irit Dinur and Kobbi Nissim. Revealing information while preserving privacy. In *Twenty-Second ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 202–210, 2003.

[DN04] Cynthia Dwork and Kobbi Nissim. Privacy-preserving datamining on vertically partitioned databases. In *CRYPTO*, volume 3152 of *LNCS*, pages 528–544, 2004.

[GB] Shafi Goldwasser and Mihir Bellare. Lecture notes on cryptography. http://www-cse.ucsd.edu/ mihir/papers/gb.html.

[Swe97] Latanya Sweeney. Weaving technology and policy together to maintain confidentiality. *Journal of Law, Medicine, and Ethics*, 25, 1997.

Appendix

A CCA-A security

In section 2.3, we described one adversary that does not have a corresponding adversary in the cryptographic literature. The game that such an adversary would attempt to win is:

1. The challenger derives a key K
2. The adversary calls $encrypt_K(m)$ for messages m.
3. The adversary presents two plaintexts, m_0 and m_1 to the challenger, who selects a bit b and returns $c = encrypt_k(m_b)$
4. The adversary performs further computations, including calls to a $decrypt_K()$ oracle, except it cannot call $decrypt_K(c)$.
5. The adversary outputs its guess for b.

A cryptosystem is said to be *CCA-A secure* if it is resilient to such an attack.

This definition is strictly weaker than CCA-2 security [CS98], and probably incomparable to CCA-1 security. A cryptosystem that is CCA-2 except that knowing a ciphertext and its decryption leaks information about how to pick a message that is easily decrypted could be CCA-A secure, but not CCA-1 secure. Likewise, a cryptosystem that is CCA-2 secure except that decryption of a ciphertext leaks information about related ciphertexts could be CCA-1 secure, but not CCA-A secure.

CCA-A security closely models the real-life situation where a ciphertext is found, and then many plaintext-ciphertext pairs, or even a limited decryption device, are found and employed to try to break the cipher. Thus, this definition warrants further study.