Differential Privacy

CS261 Fall'17, Scriber: Silvery Fu

1 What is Differential Privacy?

Differential privacy mathematically defines individuals' privacy loss when their aggregated private data are used as the subject for statistical queries. As a technique, differential privacy helps statistical databases to minimize the chances of revealing the identity of an individual while sustaining query accuracy.

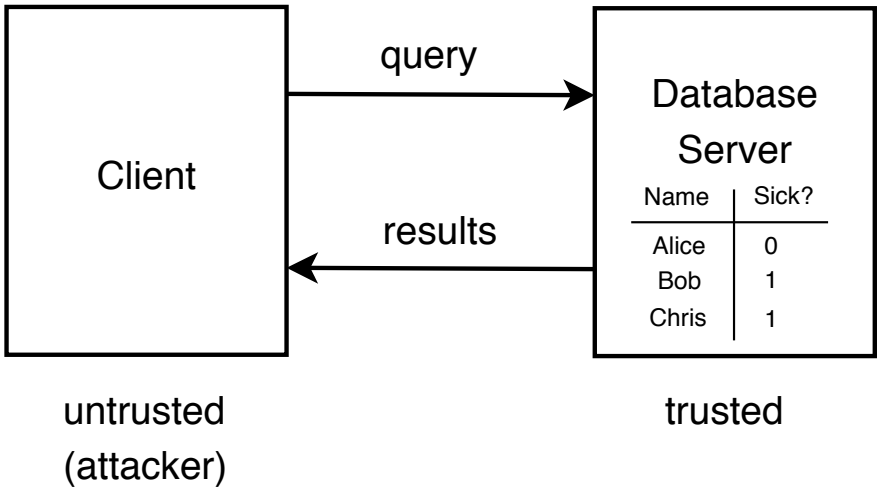

Figure 1: background and threat model

Figure 1 illustrates the interaction between a client and a database server hosting privacy-sensitive data, e.g., data about people's health conditions. Differential privacy prevents revealing the identity of an individual by **adding noise to the query results**, s.t. information about any one particular individual remain hidden. For example, a client won't be able to tell whether Chris (or any other individual) is sick or not, even if it can query about the number of sick people. A formal definition of differential privacy is given in Section 3.

1.1 Threat model

Note that the threat model is opposite to what we learned in the computation-over-encrypted-data lecture, where the database server is untrusted. Here, we assume **the database server is trusted whereas the clients are not**. The clients can query the database with aggregated/statistical

queries such as SUM, COUNT, AVG etc., *only*[1]. We assume a honest-but-curious attacker, who acts as any usual client, querying the database but not apply modifications.

How can individual privacy get compromised under this threat model? As an example, let's look at the data table in Figure 1. Suppose the attacker is able to query the database about the number of sick people. As a side information, the attacker also knows that Chris (the third entry in the table) just joined (or left) the database. By querying the database twice - once before Chris's joining and once after - and looking at the number of sick people, the attacker can figure out whether Chris is sick or healthy.

2 Why is Differential Privacy Successful?

Prior to differential privacy, there was another well-studied approach for preserving privacy, *k-anonymity*. It is "another failed blackhole" - generated lots of paper but lacked the real-world adoption. The basic idea of k-anonymity is to hide the correct answer of a query among k-1 other answers returned together by the database s.t. an attacker cannot identify the individual who is the subject of the query.

There are issues limiting the k-anonymity's practicality, however. In particular, when an attacker knows some side-information about the individual, k-anonymity can hardly help preserve privacy. For instance, the attacker can still perform the "correlation attack" on Chris as illustrated in the previous section.

On the other hand, *differential privacy is free from such correlation attack*, which is why it is successful - Apple uses it in iOS 10, Google on telemetry statistics, and US Census Bureau on communal data, for example.

How does it work? At the intuition level, differential privacy provides the nice privacy guarantee by adding noises to the returning results: instead of returning the exact number of sick people (Figure 1), the database returns the number with a random noise added. In what follows, we give the formal definition of differential privacy.

3 Formal Definition

Definition 1 *A randomized function $\mathcal{K}$ gives ϵ-differential privacy if for any two data sets D_1 and D_2 differing at most one element, for any possible set of outputs $S \in Range(\mathcal{K})$*

$$Pr[\mathcal{K}(D_1) \in S] \leqslant e^{\epsilon} Pr[\mathcal{K}(D_2) \in S] \tag{1}$$

That is, $\mathcal{K}$ returns the same result for D_1 and D_2 with roughly the same probability. Here, ϵ is the privacy factor denoting the degree of differential privacy. Next, we define the the *L1-sensitivity* of function f:

$$\Delta f = max \, \|f(D_1) - F(D_2)\| \tag{2}$$

[1] As illustrated in Section 3, even if the database server allows other types of queries, differential privacy can adapt the amount of added noises accordingly to hide an individual's identity

where the maximum is taken over all pairs of datasets D_1 and D_2 in D, and they differ in at most one element. For instance, if f is a counting function, then the $\Delta(count) = 1$; if f is to search the SSN of Bob (return SSN if found, or 0 otherwise), then the $\Delta(search\ SSN\ of\ Bob) = maximum\ number\ of\ possible\ SSNs$.

Now, given a query function f, we make it ϵ-differential private using the privacy mechanism $\mathcal{K}_f$, which is essentially the ϵ-differentially private version of f. Assuming a is the answer to query function $f(x)$, σ a fixed parameter, we have:

$$Pr[\mathcal{K}_f(x) = a] \propto e^{\frac{-\|f(x)-a\|}{\sigma}} \tag{3}$$

where $f(x)$ is the true value of query function f, and $\|f(x) - a\|$ follows the L1-sensitivity definition. This density function describes how $\mathcal{K}_f$ computes the $f(x)$ and adds noise.

4 Practical Considerations

First, we learned from the function sensitivity (eq. 2, eq. 3) that **if a function is very sensitive** (e.g., the SSN query), **then differential privacy will add lots of noise**. On the one hand, this renders the results useless; on the other hand, this protects individual user's privacy.

Second, if allowed, a client may query many times to figure out what the added noise distribution is and thus break the differential privacy guarantees. Therefore, in a practical differential private system, the system (or the data "curator") will manage the number of queries a client can launch by having a **"privacy budget."** The system either stops answering the client's query after its budge runs out or the results the client gets become noisier. For example, when receiving a query, PINQ checks whether a client has enough budget; if not, PINQ won't answer that query.

The tricky part, still, is how do we decide whether the private budget should be set for each client or set for all clients? What if clients collude with each other to figure out the added noise? The insight here is that with differential privacy, just one result should be and will be returned from the database, shared and reused among clients and queries. In other words, the database releases the results just once and does not support interactive query.

Finally, **differential privacy composes very well**. For example, if there are n independently generated randomization function, with each f_i having its privacy factor ϵ_i, then the combining function of all these functions achieves $(\sum_{i=1}^{n} \epsilon)$-differential privacy.

5 Use Case: Differentially Private Password Frequency Lists

Password Frequency List (PFLs): Given a password dataset, PFL reveals the frequency of each unique password. In this paper, the authors present a mechanism to release the PFLs - with noise added using differential privacy - to ensure that an adversary will not be able to learn anything about individual user's password from the PFL. **This holds true even if the adversary gain background knowledge about the users.**

The authors solve the key challenge of developing an efficient algorithm to sample from the exponential mechanism for integer partitions. They use a novel dynamic programming algorithm to perform approximated sampling. They demonstrate that this algorithm is efficient and adds

minimal noises to the PFL. They show that, with the differential privacy approach, **the same scientific conclusions can be drawn from the datasets despite the added noise**.